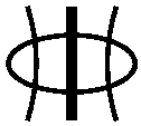
Dokumentation zum Java™Applet
Zweikörperbewegung

Michael Wünscher Sebastian Eiser

18. Oktober 2006



Technische Universität Berlin
Institut für Theoretische Physik

Zusammenfassung

Diese Dokumentation beschreibt die Bedienung und Funktion eines Java[1]-Applets, mit dem die Zweikörperbewegung in Relativkoordinaten visualisiert wird. Besonderes Augenmerk liegt dabei auf der Allgemeinheit des Potentials. Das bekannte Keplerproblem (Planetenbewegung) mit dem Gravitationspotential $-k\frac{1}{r}$ ist nur ein Spezialfall, der behandelt werden kann.
Das Java-Applet *Zweikörperbewegung* ist die Weiterentwicklung des *Kepler Applets* von *Fu-Kwun Hwang* und wurde im Rahmen des OWL Projekts *e-Module zur Veranschaulichung der Theoretischen Physik* erstellt.

[1] Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Inhaltsverzeichnis

Impressum

Institut für Theoretische Physik
Hardenbergstr. 36, Sekr. PN7-1
10623 Berlin

Projekt: *Offensive Wissen durch Lernen*
e-Module zur Veranschaulichung der Theoretischen Physik
Projektleitung: *Prof. Dr. Eckehard Schöll, PhD*
Kontakt: owl@itp.physik.tu-berlin.de

1 Einleitung

Das Java-Applet dient zur Veranschaulichung der Bewegung von zwei Körpern, die miteinander wechselwirken[2]. Einer der beiden Körper befindet sich starr im Zentrum des Applets, was durch die Verwendung von Relativkoordinaten möglich ist [3].

Was dieses Applet kann:

- Numerisch das Zweikörperproblem für nahezu beliebige Potentiale lösen
- Bahnkurven von Massepunkten zeichnen
- Bahnkurven speichern
- Anfangsbedingungen sind variabel
- Das zweite Kepler'sche Gesetz (Flächensatz) visualisieren
- Effektive Potentiale zeichnen und die aktuellen Position der Masse anzeigen
- Den Harmonischen Oszillator (und weitere vordefinierte Potentiale) in 2 Dimensionen visualisieren

Was das Applet nicht kann:

- Dreidimensionale Darstellungen
- beliebig hohe Energien als Anfangsbedingungen akzeptieren
- der numerische Integrator (*Runge-Kutta-4* Verfahren) kann bei zu hohen Geschwindigkeiten Fehler erzeugen

Der bekannteste Fall des *Zweikörper-Problems* (und auch die Voreinstellung beim Start des Applets) ist die Planetenbewegung, auch Keplerproblem genannt. Bei der Planetenbewegung wirkt die Gravitationskraft, die mit $\frac{1}{r^2}$ (r Abstand zum Kraftursprung) abfällt, also ein $-\frac{1}{r}$ Potential aufweist.
Die Besonderheit dieses Applets besteht in der Möglichkeit, dieses Potential (also auch die Wechselwirkungskraft) mit einer Formel nahezu beliebig zu verändern. Damit können nun verschiedene physikalische oder auch hypothetische Potentiale simuliert werden. Beispiele dafür *Harmonischer Oszillator* oder das *Coulomb* Potential (nur klassisch). Aber auch Mischformen von verschiedenen Potentialen (z.B. $V(r) = -r^{-1} + r^2$), die von theoretischem Interesse sind, können visualisiert und betrachtet werden. Man kann auf diese Weise einfach sehen, dass z.B. die Bahn eines Planeten nicht mehr elliptisch ist, sobald kein reines Gravitationspotential vorliegt (siehe Abbildung 1).

Verändert man die Koeffizienten bzw. Potenzen der im oberen Abschnitt des Fensters (Abbildung 2) angezeigten Formel, lässt sich das Potential nahezu beliebig formen. Natürlich ergeben nicht alle möglichen Kombinationen eine brauchbare Simulation, deswegen sind schon einige Voreinstellungen im *Pull-Down Menü* wählbar, die als Anhaltspunkte dienen können.
Achtung: hohe Potentiale (entsprechen hohen Energien) können vom Integrator nicht gehandhabt werden, da dann die Geschwindigkeiten zu groß werden. Das Gleiche gilt für effektive Potentiale, deren Minima sehr nahe im Ursprung liegen.

[2] Ausnahme: Potential wird konstant gewählt.

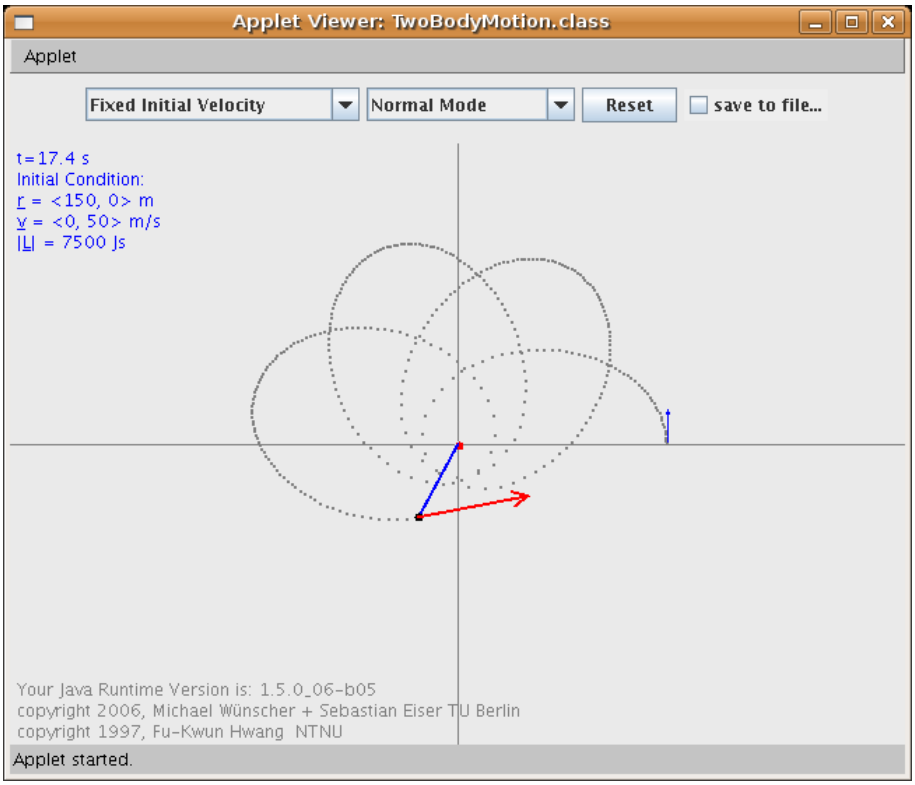


Abbildung 1: Screenshot der Anwendung (Hauptfenster).

2 Bedienung des Applets

Nach dem Starten des Applets erscheinen auf dem Bildschirm zwei Fenster, die im folgenden erläutert werden.

2.1 Benutzeroberfläche

Hauptfenster

Das Hauptfenster ist in zwei Bereiche eingeteilt. Oben befinden sich die Bedienungselemente mit den zwei Auswahllisten, dem ,,Reset“-Knopf und dem ,,Speichern“-H”ackchen, die weiter unten erläutert werden. Im darunterliegenden Zeichenbereich wird die graphische Darstellung der Trajektorie des zweiten Teilchens dargestellt und es lassen sich die Anfangsbedingungen mit der Maus vorgeben.
Die **Trajektorie** wird durch eine graue gestrichelte Linie dargestellt und wird als Spur des zweiten Teilchens gezeichnet.
Im Zentrum des **Koordinatensystems** (schwarz) befindet sich das **Zentralteilchen** (rot). Von ihm wird der **Verbindungsvektor** (blau) zum zweiten Teilchen (schwarzer Punkt) gezeichnet. Vom zweiten Teilchen aus wird der **Geschwindiglkeitsvektor** (roter Pfeil) gezeichnet, der die Geschwindigkeit in Richtung und Betrag im Relativkoordinatensystem darstellt.
Die **Anfangsbedingung** kann durch Verschieben und ”Andern der Länge des blauen Vektorpfeils im ersten Quadranten mit der Maus eingestellt werden, dazu wird dieser mit der linken Maustaste angeklickt und mit gedrückt-gehaltener Maus auf der x-Achse verschoben werden. Auf diese Weise wird der Anfangsradius geändert. Wenn in der Aus-

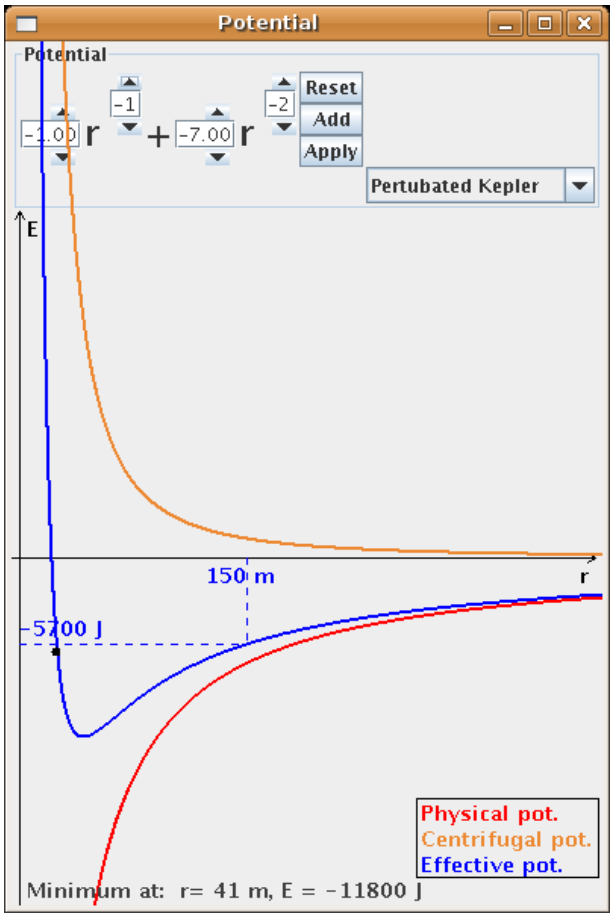
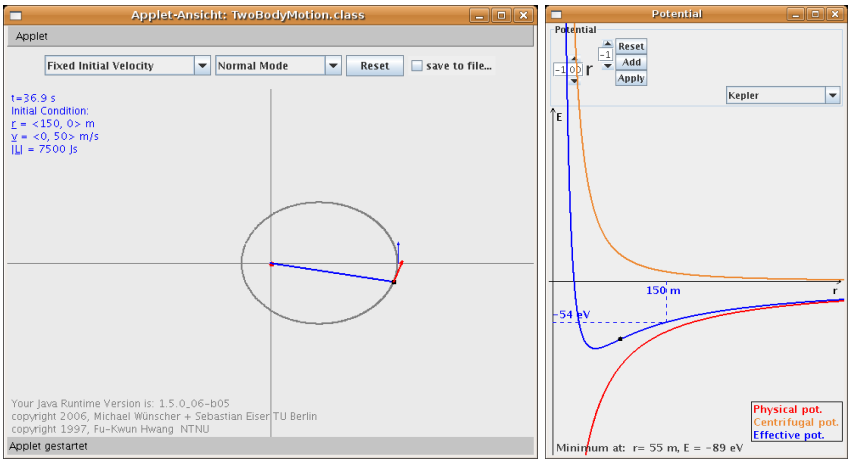


Abbildung 2: Potentialfenster

wahlliste für die Anfangsbedingung (siehe unten) der Punkt ,,arbitrary initial conditions" (bzw. ,,beliebig" in der deutschen Version) ausgewählt ist, kann durch Bewegen der Maus entlang der y-Achse auch die Länge des Vektors und damit die Anfangsgeschwindigkeit geändert werden.
Die Animation kann durch Drücken der **reche Maustaste** im Zeichenbereich angehalten werden und durch nochmaliges Drücken wieder in Gang gesetzt werden.

Erläuterung der Bedienelemente:

- **Auswahlliste für die Anfangebedingung**
  - Fix initial velocity - einstellen der Anfangsbedingung ohne Änderung der Anfangsgeschwindigkeit
  - Fix angular momentum - einstellen der Anfangsbedingung ohne Änderung des Drehmomentes
  - arbitrary initial condition - beliebige Anfangsbedingungen möglich im ersten Quadranten mit einer Anfangsgeschwindigkeit in z-Richtung
- **Auswahlliste für die graphische Darstellung**
  - Normal Mode - normale Visualisierung



(a) **Hauptfenster** (b) **Potentialfenster**

Abbildung 3: (a) Zeigt die Trajektorie mit einem Teilchen fest im Ursprung (b) Zeigt das effektive Potential, das Drehimpuslpotential und das Zentralpotential

  - Law of equal area - zusätzlich anzeigen, welche Fläche in einer bestimmten Zeit überstrichen wird

- **„Reset“-Knopf** Durch Drücken des ,,Reset“-Knopf werden die Anfangsbedingungen und das Potential für das Kepler-Problem eingestellt.

- **„save to file“-Haken** Nach dem Setzen des Haken durch Anklicken des Blockes wird der Speicherort für eine Datei abgefragt, in der die Daten der numerischen Simulation abgelegt werden sollen. Das Speichern der Daten wird solange durchgeführt, bis der Haken wieder entfernt wird.

Potentialfenster

Das Potentialfenster ist wie das Hauptfenster in zwei Bereiche eingeteilt. Den oberen Bereich der Bedienelemente und den unteren **Zeichenbereich** auf dem die drei Potentiale (effektives Potential, Drehimpulspotential und Zentralpotential) gezeichnet werden. Diese sind passend zu den Anfangsbedingungen dargestellt, die im Hauptfenster eingestellt wurden. Durch gestrichelte Linien wird der Anfangsradius und die zugehörige Anfangsenergie eingezeichnet und mit den entsprechenden Werten gekennzeichnet. Eine Legende für die Potentiale befindet sich unten rechts. Links neben dieser werden die Koordinaten des Potentialsminimums vom effektiven Potential ausgegeben. Falls es kein Minimum gibt wird ein Hinweistext ausgegeben. In diesem Fall kann die ganze Bewegung nicht vollständig im Hauptfenster dargestellt werden und es sollten gegebenenfalls die Vorfaktoren für das Potential verändert werden (siehe unten).

Zum **Verändern des Zentralpotentials** gibt es im Bedienbereich verschiedene Elemente. Die Formel wird aus mindestens einem Paar bestehend aus **Vorfaktor** und **Potenz** vom **Radius r** dargestellt. Sowohl der Vorfaktor als auch die Potenz können durch Drücken auf die Pfeile über und unter der Zahl verändert werden. Dies kann auch durch die direkte Eingabe eines Zahlenwertes geschehen. Nach der Änderung des Potentials muss diese durch den ,,Apply“-Knopf bestätigt werden. Das Zentralpotential kann aus mehreren solchen Termen zusammengesetzt werden, die durch den ,,Add“-Knopf hinzugefügt werden

können.
Bedienelemente:

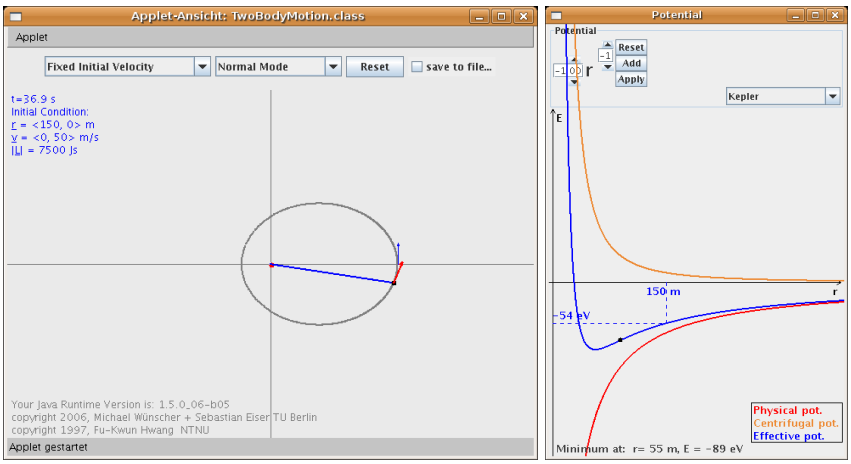
- **„Reset“-Knopf** - Ale Potentialterme bis auf den ersten werden gelöscht
- **„Add“-Knopf** - Anfügen eines neuen Potentialterms
- **„Apply“-Knopf** - Bestätigen von Änderungen am Potential und erneutes Starten der Simulation mit den eingestellten Anfangsbedingungen
- **Auswahlliste für die Presets** - Auswählen eines vorgegebenen Potentials wie zum Beispiel das Kepler Potential oder das Potential des harmonischen Oszillators

2.2 Tipps für Vorführung

Bei der Vorführung des Applets sind die **Presets** sehr hilfreich. Diese befinden sich im *Pull Down Menus* des Potentialfensters. Es stehen verschiedene vorgefertigte Potentiale mit geeignet gewählten Vorfaktoren bereit, die man einfach auswählen kann. Nach der Auswahl wird das entsprechende Potential geladen, dargestellt und die Animation wird neu gestartet.
Falls die Animation mit veränderten Parametern nicht mehr richtig laufen sollte, kann das Applet entweder durch das Laden eines Presets oder durch Drücken des ,,Reset“-Knopfs im Hauptfenster wieder in einen definierten Zustand versetzt werden.
Falls das Potentialfenster nicht mehr geöffnet ist, kann man dies durch Drücken des ,,Reset“-Knopf im Hauptfenster wieder öffnen. Achtung! Dabei werden das eingestellte Potential und die Anfangsbedingungen zurückgesetzt.

Vorgefertigte Potentiale:

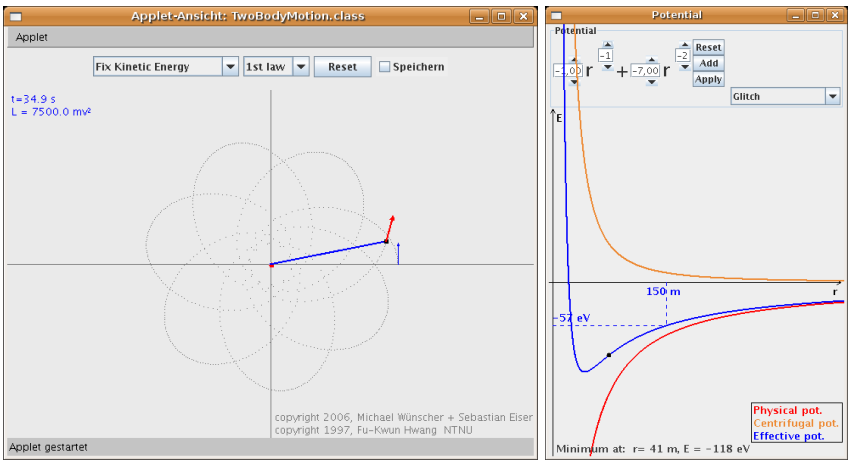
- **Kepler potential** $V(r) = -1 \cdot 10^6 r^{-13}$ (Abb. 4)



(a) **Hauptfenster** (b) **Potentialfenster**

Abbildung 4: *Kepler* - Dient der Veranschaulichung der Bewegung der Erde um die Sonne

- **Modified Kepler** $V(r) = -1 \cdot 10^6 r^{-1} - 7 \cdot 10^6 r^{-2}$ (Abb. 5)

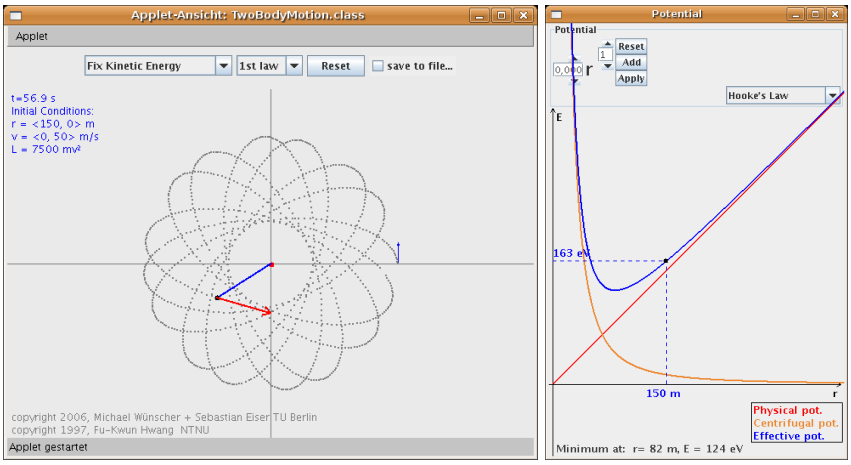


(a) **Hauptfenster** (b) **Potentialfenster**

Abbildung 5: *Kepler modifiziert* - Durch Veränderung des Potentials ergibt sich eine Periheldrehung

[3] Die Vorfaktoren im Programm werden mit 10^6 multipliziert, damit sie zu den Bildschirmkoordinaten passen.

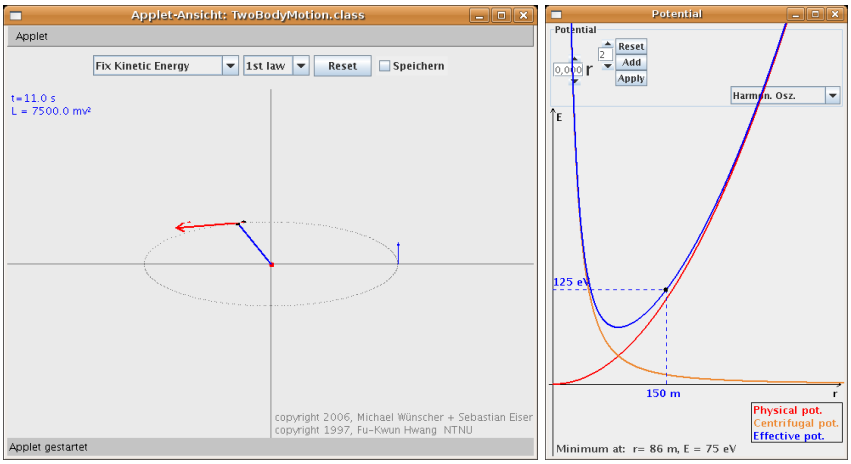
- **Constant force** $V(r) = 100 \cdot r^1$ (Abb. 6)



(a) **Hauptfenster** (b) **Potentialfenster**

Abbildung 6: *Constant force* - Veranschaulichung des Drehimpulsanteils

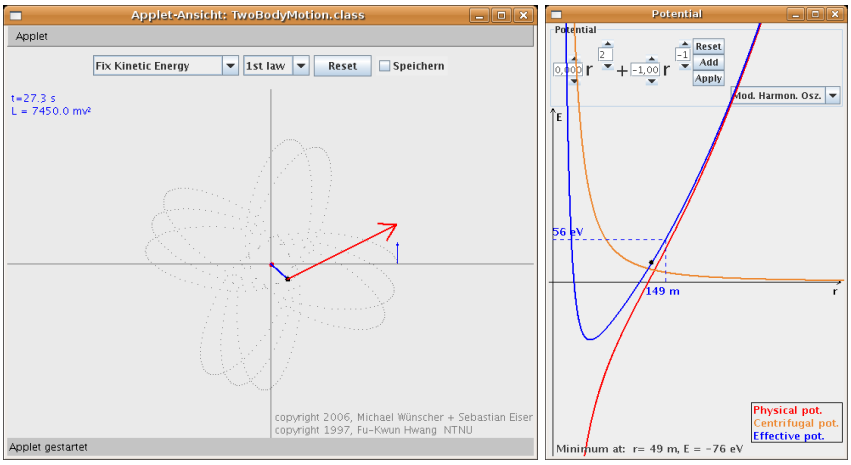
- **Harmonic oscillator** $V(r) = 0,5 \cdot r^2$ (Abb. 7)



(a) **Hauptfenster** (b) **Potentialfenster**

Abbildung 7: *Harmonischer oscillator* - Zwei gekoppelte Pendel jeweils in x- und in y-Richtung

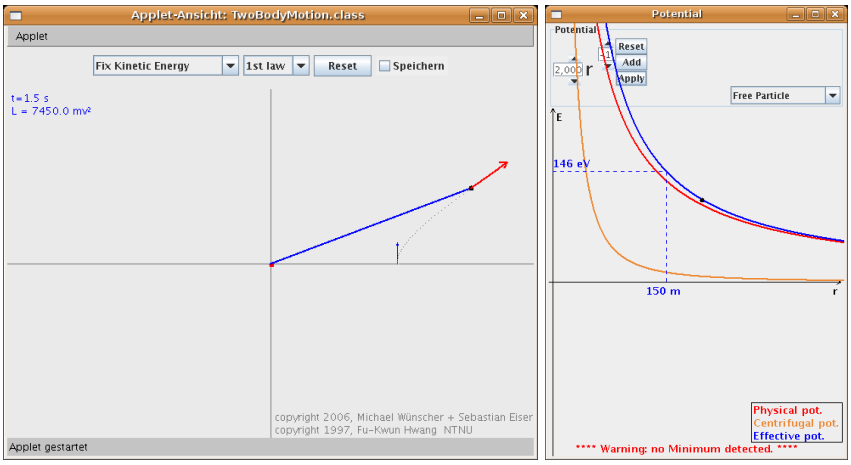
- **Modified harmonic oscillator** $V(r) = 0,5 \cdot r^2 - 1 \cdot 10^6 r^{-1}$ (Abb. 8)



(a) **Hauptfenster** (b) **Potentialfenster**

Abbildung 8: *modifizierter Harmonischer Oszillator* - Zwei gekoppelte Pendel überlagert mit dem Kepler Potential

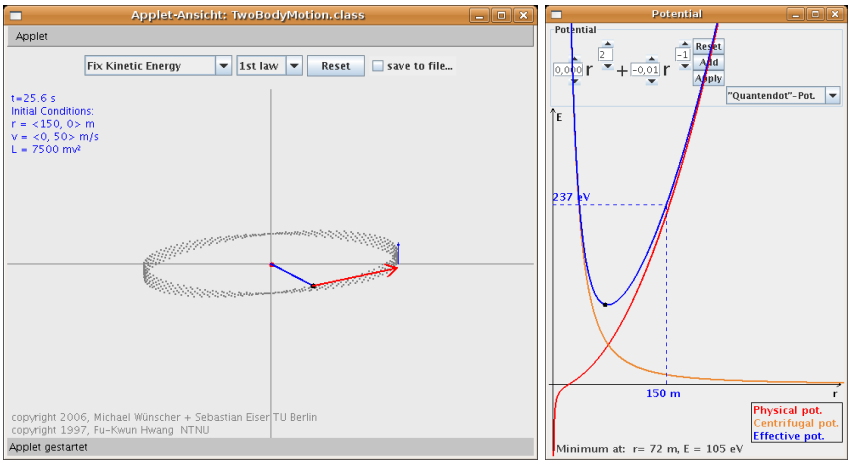
- **Repulsive potential** $V(r) = 2 \cdot 10^6 r^{-1}$ (Abb. 9)



(a) **Hauptfenster** (b) **Potentialfenster**

Abbildung 9: *Abstoßendes Potential* - Das zweite Teilchen entfernt sich vom Kraftzentrum

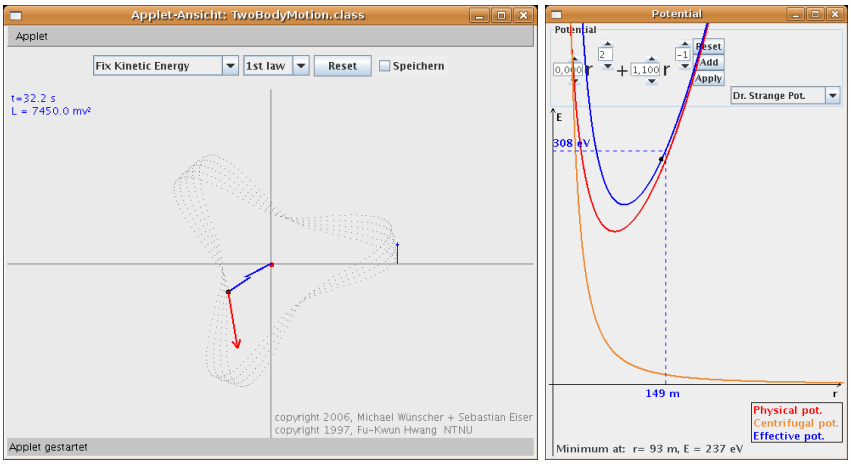
- **„Quantendot“ potential** $V(r) = 1 \cdot r^2 - 1 \cdot 10^4 r^{-1}$ (Abb. 10)



(a) **Hauptfenster** (b) **Potentialfenster**

Abbildung 10: *„Quantendot“ potential* - Das richtige Verhalten??

- **Preset 1** $V(r) = 1 \cdot r^2 + 10^6 \cdot r^{-1}$ (Abb. 11)



(a) **Hauptfenster** (b) **Potentialfenster**

Abbildung 11: *Preset 1*

2.3 Stand alone Version

Das Applet läßt sich auf verschiedene Arten starten. Zum einen kann es durch Öffnen im Browser sichtbar gemacht werden. Eine andere Möglichkeit ist das Starten des Applet mit dem Java Appletviewer. Dazu gibt man auf der Konsole den folgenden Befehlt ein:

```
appletviewer <Adresse oder Pfad der HTML-Datei des Applets>
```

Das Applet kann also mit Hilfe des Appletviewers auch auf PCs ohne Internetanschluß benutzt werden. Dazu sind nur zwei Dateien nötig. Diese sind zum einen das Applet in einer *Jar-Datei* ,,TwoBodyMotion.jar" und die HTML-Datei zum Aufruf des Applets aus dieser Jar-Datei. Die beiden Dateien müssen in einen lokalen Ordner kopiert werden und mit dem Appletviewer unter Angabe des Pfades der HTML-Datei gestartet werden.

3 Programm

Das Programm wurde für Java 1.5 (5) geschrieben und getestet. Es ist allerdings auch kompatibel zu Java 1.4 und vermutlich auch früheren Versionen. Bei Konflikten sollte man dennoch die neuerste Version installiert haben. Im Hauptfenster des Applets wird die eigene Version dargestellt.

Wer die Quelltexte untersuchen will, findet vielleicht folgende Webseiten interessant:

- Die Java 1.5 Dokumentation

  `(http://java.sun.com/j2se/1.5.0/docs/api/)`

- Tutorialsammlung zu Java

  `(http://java.sun.com/docs/books/tutorial/uiswing/TOC.html)`

- Mehr Beispiele zu 2D Programmierung

  `(http://java.sun.com/products/java-media/2D/samples/suite/)`

3.1 Programmstruktur

Grundsätzlich werden von dem Programm zwei Integrationen durchgeführt, da es sich um eine Bewegung in der Ebene handelt. Eine für die Beschleunigung in *X-Richtung* (horizontal) und eine in *Y-Richtung*. Dabei wird jeweils von dem gewählten Potential der Gradient gebildet und die vom aktuellen Ort abhängige Beschleunigung bestimmt. Diese, etwa alle 50 Millisekunden aufgerufene Operation wird von einem separaten *Thread* geringer Priorität ausgeführt.

Kurzbeschreibung der Klassen

Eine genauere Erläuterung befindet sich im Quelltext der Anwendung.

- **TwoBodyMotion Class:** Diese (von *java.applet.Applet* abgeleitete) Klasse ist die Hauptklasse. Sie übernimmt die Initialisierungen, steuert den Integrator-Thread, überwacht Mausevents und reagiert darauf. Der Integrator-Thread ist ebenfalls Teil dieser Klasse und wendet zur numerischen Integration das Runge-Kutta Verfahren (4.Ordnung) an, mit dem die Differentialgleichungen gelöst werden. Des Weiteren löst dieser das Neuzeichnen der Fenster nach abgeschlossener Berechnung aus.
- **PotentialFenster Class:** Das zweite Fenster der Anwendung, in dem die Potentialformel und die daraus generierte Kurve dargestellt werden, ist von der Java Klasse *JFrame* abgeleitet. Neben den nötigen Zeichenarbeiten reagiert es auf Events, die im Potentialfenster auftreten.
- **potential Class:** Diese Klasse speichert die Formel des aktuellen Potentials in einem Array vom Typ double ab. Sie enthält Funktionen zum Erstellen und Modifizieren des Potentials, sowie dessen Gradientenbildung.
- **PotentialPanel Class:** Die von *JPanel* abgeleitete Klasse zeichnet, erstellt und verwaltet das im Potentialfenster oben angezeigte Formelfeld mit integriertem *Pulldown-Menü*. Es erlaubt das Eingeben des Polynoms, das zur Potentialangabe verwendet wird. Es enthält (maximal 10) FormulaPanels (siehe unten) und enthält die *Potential-Presets*. Von außen sind *ActionListener* zugänglich, die den *Apply Button* und die Presets überwachen. Die Funktion *getCoefficients()* erlaubt das Zugreifen auf die aktuellen Koeffizienten und Potenzen in der Formel.
- **FormulaPanel Class:** Ein FormulaPanel (von JPanel abgeleitet) besteht aus zwei upDownPanels (siehe unten) und einem Label für das Anzeigen des Buchstabens *r* und ordnet diese Objekte so an, dass ein upDownPanel den Koeffizienten und das andere die Potenz von *r* angibt.
- **upDownPanel Class:** Diese (von JPanel abgeleitet) Klasse besteht aus einem Textfeld und einem darüber, sowie darunter angeordnetem Pfeil. Es erlaubt selbstständig die Manipulation des Zahlenwerts im Textfeld.
- **ArrowIcon Class:** Ein selbsterstellter Button mit Pfeilsymbol.
- **Gravity Class:** Diese von rk4 abgeleitete Klasse enthält ein Objekt der potential Klasse und implementiert die Ableitungsfunktion, die von der Klasse rk4 benutzt wird. Diese Klasse ist die Schnittstelle zwischen unsere Potentialklasse und der Runge-Kutta Klasse. Von dieser Klasse wird ein Objekt *particle* im Hauptprogramm benutzt, um einen Integrationsschritt zu berechnen. Diese wird durch den Integrator-Thread ausgelöst. Dabei wird jeder Anzeigeschritt mit 200 Zwischenschritten berechnet.
- **rk4 Class:** Durch die Klasse wird das herkömmlichen Runge-Kutta Verfahren 4. Ordnung implementiert und mit dessen Hilfe die Differentialgleichung numerisch gelöst. Zu Beachten ist, dass dies eine abstrakte Klasse ist. Zur Benutzung muss erst eine abgeleitete Klasse die Ableitung durch die Funktion *derivs (double t,double y[],double dydx[])* implementiert werden.

3.2 Übersetzen des Quellcodes

Für die Entwicklung des Applets haben wir die Entwicklungsumgebung ,,Eclipse” *(Version 3.1)* verwendet.

Zum Übersetzen des Programms werden die Quelltextdateien (*.java) aus dem Tar-Gz-Archiv in ein Arbeitsverzeichnis entpackt. Danach wird Eclipse gestartet. Zum Anlegen eines neue Projektes wird $file \rightarrow new \rightarrow project$ gewählt. Als nächstes trägt man den Namen des Projektes ein und wählt das Arbeitsverzeichnis im Punkt ,,Create project from existing sources" aus. Jetzt sollte das eben angelegte Projekt im Package-Explorer auftauchen. Die Quelltexte sollten vom automatischen (im Hintergrund laufenden) Compiler in die entsprechenden Class-Dateien übersetzt worden sein.
Zum Starten des Applets muss zuerst die Hauptklasse ,,TowBodyMotion.java" geöffnet werden, da sich in dieser Datei das Hauptprogramm befindet. Dazu wählt man im Package-Explorer den Unterpunkt ,,default" die Datei aus. Im Menü ,,Run" muss man für den ersten Start den Punkt ,,Run..." auswählen. Es öffnet sich ein Konfigurationsdialog, in dem die Parameter width=620 und heigth=400 im entsprechenden Unter-Tab gesetzt werden müssen. Nach dem Bestätigen mit dem Button ,,Run" öffnet sich das Applet.
Falls es Probleme mit dem Kompilieren gibt, könnte dies an einer falschen Java version liegen. Diese kann im Unter-Tab ,,JRE" vom Run-Dialog einstellen, falls weitere Java-Engines installiert sind. Diese können im Preference-Dialog, zu finden im Menü $window \rightarrow preference$, unter $Java \rightarrow Installed\,JRE \rightarrow Add$ hinzugefügt werden. Unter Ubuntu 6.06 befinden sich die JRE's im Verzeichnis /usr/lib/jvm (Sun-Java) oder /usr/lib/j2se (Blackdown-Java). Dies läßt sich gegebenenfalls durch die Eingabe von ,,locate javac" auf der Konsole herausfinden.
Für die Veröffentlichung des Programms ist es hilfreich, alle Klassen in einem Kontainer, der Jar-Datei, aufzubewahren. Dazu wählt man in Eclipse unter $file \rightarrow export$ den Punkt ,,Jar File" aus. Nach der Selektion des default packages aus dem Projekt gibt man noch den Speicherort an. Die restlichen Einstellungen reichen im Normalfall aus.
Da wir ein paar Funktionen benutzen, die im Browser nicht ohne Zertifizierung bzw. Signierung erlaubt sind (darunter das Schreiben von Dateien und bestimmte Fenster-Funktionen), muss die Jar-Datei noch signiert werden. Dazu ist als erstes ein Schlüssel notwendig, den man einmal erzeugen muss. Diesen erzeugt man mit dem Programm ,, keytool". Der Aufruf könnte wie folgt aussehen:

```
keytool -genkey -alias yourAlias
```

Danach werden von dem Programm noch weitere Informationen wie z.B. das Passwort für die lokale Schlüsseldatenbank, der Name und die Institution erfragt. Details sind im Internet leicht zu finden.
Das eigentliche Signieren mit einem Zertifikat wird mit dem Program ,,jarsigner" erledigt. Dies sollte sich mit dem Aufruf

```
jarsigner jar-datei yourAlias
```

erledigen lassen. Vom Programm wird nur noch das Passwort für die Schlüsseldatenbank abgefragt.

3.3 Veröffentlichen im Internet

Bei den Quelldateien ist eine HTML-Datei für die Veröffentlichung mit enthalten. Diese sollte man am besten mit Javascript in einem Pop-up-Fenster, passend zur Größe des Hauptfensters, öffnen, da nicht jede Java-Engine die Option ,,AlwaysOnTop()" kennt, womit das Potentialfenster auch beim Verändern der Anfangsbedingungen sichtbar bleibt. Ansonsten bleibt das Potentialfenster hinter dem Browserfenster.

Die HTML-Datei könnte wie folgt aussehen:

```
<html>
<body>
<applet ARCHIVE="TwoBodyMotion.jar"
  code="TwoBodyMotion.class" width="640" height="480">
<param name="L1" value="Normale Visualisierung">
<param name="L2" value="Flaechensatz">

<param name="InitialCondition" value="Anfangsbedingung">
<param name="Reset" value="Reset">
<param name="C1" value="Feste Anfangsgeschwindigkeit">
<param name="C2" value="Fester Drehimpuls">
<param name="C3" value="beliebig">

<param name="L3" value="speichern...">
<param name="L4" value="energy">
<param name="MSG1" value="Anfangsbedingung">
<param name="MSG2" value="mouse X,Y">
<param name="T" value="Period=">
<param name="R" value="r=">
<param name="invalid" value="invalid">
</applet>
</body></html>
```

Dabei brauchen die param-tags nicht vorhanden sein; sie dienen in diesem Fall nur zum Übersetzen des Applets ins Deutsche. Ohne diese Tags erscheint das Applet in englischer Sprache.

Die Javascript-Funktion zum Öffnen eines Pop-Ups könnte wie folgt aussehen:

```
<script language="javascript">
function zeige() {
window.open("index2.html", width=640, height=550);
}
</script>
```

und läßt sich zum Beispiel an einen Link binden.

```
<a href="" onclick="zeige()">Hier klicken</a>, um das <B>neue</B> Applett zu starten.
```

Nach dem Betätigen des Links öffnet sich ein Pop-Up-Fenster mit dem Applet, wenn in der Datei ,,index2.html" der HTML-Code zum Starten des Applets enthalten ist.

3.4 Technische Voraussetzungen

Wir empfehlen das Applet auf einem Rechner mit mindestens 600 MHz und 64 MB RAM zu verwenden, da Java einen Just-in-Time Compiler verwendet und vergleichsweise aufwendige Rechenoperationen durchgeführt werden. Der Bildschirm muss eine Auflösung von mindestens 800×600 verwenden und sollte 16.6 Millionen Farben unterstützen.

4 Theorie (kurz)

4.1 Beschreibung mittels Differentialgleichung

Im Folgenden beschränken wir uns auf die Lösung des klassischen Zweikörperproblems mit Zentralpotential (siehe [2][3]). Die Newtonschen Bewegungsgleichungen lauten

$$\begin{aligned} m_1\ddot{\underline{r}}_1 &= -\bigtriangledown_{\underline{r}_1} V(|\underline{r}_1 - \underline{r}_2|) \\ m_2\ddot{\underline{r}}_2 &= -\bigtriangledown_{\underline{r}_2} V(|\underline{r}_2 - \underline{r}_1|) \end{aligned}$$

wobei das Potential im Applet die Form

$$V(r) = V(|\underline{r}_1 - \underline{r}_2|) = -G \cdot \frac{m_1 m_2}{|\underline{r}_1 - \underline{r}_2|^\alpha}$$

hat und r der Betrag des Abstandes der zwei Körper ist.
Die DGL für $\alpha = 1$ (Kepler Problem) lautet dann

$$\begin{aligned} m_1\ddot{\underline{r}}_1 &= -G\frac{m_1 m_2}{|\underline{r}_1 - \underline{r}_2|^3}(\underline{r}_1 - \underline{r}_2) \\ m_2\ddot{\underline{r}}_2 &= -G\frac{m_1 m_2}{|\underline{r}_2 - \underline{r}_1|^3}(\underline{r}_2 - \underline{r}_1) \end{aligned}$$

Diese wird durch den Integrator im Runge-Kutter-Verfahren mit den gegebenen Anfangebedingungen gelöst (siehe 4.2).
Analytisch kann man diese Problem ebenfalls lösen. Das DGL lässt sich in Relativkoordinaten reduzieren zu

$$\begin{aligned} (I)\ m_2 m_1\ddot{\underline{r}}_1 &= -m_2 \bigtriangledown_{\underline{r}_1} V(|\underline{r}_1 - \underline{r}_2|) \\ (II)\ m_1 m_2\ddot{\underline{r}}_2 &= -m_1 \bigtriangledown_{\underline{r}_2} V(|\underline{r}_2 - \underline{r}_1|) \\ (I-II):\ m_2 m_1(\ddot{\underline{r}}_1 - \ddot{\underline{r}}_2) &= -m_2 \bigtriangledown_{\underline{r}_1} V(|\underline{r}_1 - \underline{r}_2|) + m_1 \bigtriangledown_{\underline{r}_2} V(|\underline{r}_2 - \underline{r}_1|) \\ \Rightarrow\ \mu\ddot{\underline{r}} &= -\bigtriangledown_{\underline{r}} V(r) \end{aligned}$$

mit $\underline{r} = \underline{r}_2 - \underline{r}_1$, $r = |\underline{r}_2 - \underline{r}_1| = |\underline{r}_1 - \underline{r}_2|$ und der reduzierten Masse $\mu = \frac{m_1 m_2}{m_1 + m_2}$. Wir leiten zuerst den Drehimpulserhaltungssatz ab. Dieser gilt für allgemeine Zentralpotentiale, weshalb wir dies in Relativkoordinaten tun.

$$\begin{aligned} \underline{L} &= \underline{r} \times p = \mu \underline{r} \times \dot{\underline{r}} \\ \dot{\underline{L}} &= \mu(\dot{\underline{r}} \times \dot{\underline{r}}) \\ &= \mu(\underbrace{\dot{\underline{r}} \times \dot{\underline{r}}}_{=0, da\ \dot{\underline{r}} \parallel \dot{\underline{r}}} + \underline{r} \times \ddot{\underline{r}}) \\ &= \underbrace{\underline{r} \times -\bigtriangledown_{\underline{r}} V(r)}_{=0, da\ V(r)\ zentralsymmetrisch \Leftrightarrow \bigtriangledown_{\underline{r}} V(r) = |\bigtriangledown_{\underline{r}} V(r)| e_r} = 0 \\ \Rightarrow \underline{L} &= const \end{aligned}$$

Die Bewegung findet also nur in der Ebene senktrecht zum Drehimpuls $\underline{L}$ statt. Ohne Beschränkung der Allgemeinheit legen wir den Drehimpuls $\underline{L} = L \cdot e_z$ in z-Richtung. Das hat für die numerische Lösung die Folge, dass wir die Bewegung nur in der xy-Ebene berechnen brauchen. Des Weiteren verlieren wir keine Information durch die Darstellung der Bewegung in der xy-Ebene. Erwähnt sei noch, dass es auch durchaus eine Bewegung des hier nicht betrachteten Schwerpunktes in z-Richtung geben kann.

Um die Bewegung schon an ihrem Potential zu erkennen führen wir ein effektives Potential V_{eff} ein, denn dieses zeigt alle Merkmale der Bewegung auf. Dazu nutzen wir den Energieerhaltungssatz und den Drehimpuls, die wir in Polarkoordinaten umschreiben.

$$\begin{aligned} L &= \mu \underline{r} \times \dot{\underline{r}} = -\mu r^2 \dot{\phi}^2 \underbrace{e_\vartheta}_{,da\ \vartheta=\frac{\pi}{2}} = \mu r^2 \dot{\phi}^2 e_z \\ E &= \frac{1}{2}\mu \dot{\underline{r}}^2 + V(r) = \frac{\mu}{2}(\dot{r}^2 + r^2\dot{\phi}^2) + V(r) \\ \Rightarrow E &= \frac{\mu}{2}\dot{r}^2 + \frac{L^2}{2\mu r^2} + V(r) = \frac{\mu}{2}\dot{r}^2 + V_{eff}(r) \end{aligned}$$

Das effektive Potential besitzt, wenn ein gebundener Zustand vorliegt, ein Potentialminimum. Die Schnittpunkte mit der Energie ergeben die Umkehrpunkte bzw. den minimalen und maximalen Radius der Bewegung. Zwischen diesen Punkten findet die Bewegung statt.
Die gezeigten Berechnungen lassen sich leicht für ein zusammengesetztes Potential mit N Termen wie es im Applet simuliert wird, nachvollziehen. Das simulierte Potential lautet

$$V(r) = -\sum_{i=1}^{N} G_i M_i \cdot r^{\alpha_i}$$

mit dem Vorfaktor $G_i M_i$ und der Ganzzahligen Potenz α_i. Im Applet wird noch eine Skalierung zur Anpassung an die Bildschirmauflösung benutzt. Dazu wird jeder Vorfaktor $G_i M_i$ mit dem Wert 10^6 multipliziert, dieser Wert ist so gewählt, dass das Kepler Problem gut dargestellt wird. Des Weiteren ist in diesen Vorfaktoren die Masse schon enthalten.

4.2 Numerische Integration

Für die numerische Integration sei auf [1] verwiesen.
Die DGL (2.Ordnung) wird in ein DGL System erster Ordnung überführt. Es lautet

$$\begin{aligned} \dot{r}_x &= v_x \\ \dot{r}_y &= v_y \\ \dot{v}_x &= -(\bigtriangledown_r V(r))_x \\ \dot{v}_y &= -(\bigtriangledown_r V(r))_y \end{aligned}$$

und wird mit den variablen Anfangsbedingungen $\underline{r} = (150, 0)$ und $\underline{v} = (0, 50)$ gelöst. Für die Lösung implementieren wir die Berechnung des Gradientens des allgemeinen Potential nach der folgenden Regel für eine Komponente $i \in \{x, y, z\}$

$$(\bigtriangledown_r V(r))_j = \underline{r}_j \sum_{i=1}^{N} G_i M_i \cdot r^{\alpha_i - 2}$$

Mit dieser Regel können wir also die Ableitungsfunktion implementieren, die in der Literatur meist mit $f(x, \underline{y})$ bezeichnet wird. Hier ist die Variable x die Zeit und der Vektor $\underline{y} = (r_x, r_y, v_x, v_y)$ unser Gleichungssystem. In unserem Fall ist die Ableitung nicht explizit zeitabhängig und die Geschwindigkeitskomponenten hängen nicht von der Geschwindigkeit ab.

Damit lässt sich das Gleichungssystem dann wie folgt lösen (Achtung, der Index i bezeichnet jetzt den diskreten i-ten Zeitschritt)

$$\begin{aligned}
\underline{Y}_{i+1} &= \underline{Y}_i + h_i(\frac{1}{6}\underline{k}_1 + \frac{1}{3}\underline{k}_2 + \frac{1}{3}\underline{k}_3 + \frac{1}{6}\underline{k}_4) \;\; with \\
\underline{k}_1 &= f(x_i, \underline{Y}_i) \\
\underline{k}_2 &= f(x_i + \frac{h_i}{2}, \underline{Y}_i + h_i\frac{\underline{k}_1}{2}) \\
\underline{k}_3 &= f(x_i + \frac{h_i}{2}, \underline{Y}_i + h_i\frac{\underline{k}_2}{2}) \\
\underline{k}_4 &= f(x_i + h_i, \underline{Y}_i + h_i\underline{k}_3))
\end{aligned}$$

wobei die Variablen $\underline{r}$ und $\underline{v}$ zu dem diskretisierten Vierervektor $\underline{Y_i} = (r_x, r_y, v_x, v_y)_i$ zusammengefasst wurden und mit der Variablen h die Schrittweite eingestellt wird.

A Quelltexte

Listing 1: Hauptklasse des Applets

-5mm

```
/**
* This file (TwoBodyMotion.java) is part of the Two Body Motion project.
*
* required files: ArrowIconSeb.java, upDownPanel.java, FormulaPanel.java, potential. ...
    java,
*                                 PotentialPanel.java, Gravity.java, PotentialFenster. ...
    java
*
* used by: TwoBodyMotion.java
*
* ---------------------------------------------------------
* Das ist die Implementierung des Zweikoerperproblems als Applet. Es initialisiert die
* beiden Fenster und startet die Simulation. Es beinhaltet das Hauptprogramm und ...
    startet
* die Threads zur numerischen Integration
*

*
* @author Fu-Kwun Hwang, Michael Wuenscher, Sebastian Eiser
*/

import java.awt.*;
import java.io.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.event.ChangeEvent;

/**
* @author Fu-Kwun Hwang, Michael Wuenscher, Sebastian Eiser
*/
public class TwoBodyMotion extends java.applet.Applet
                implements Runnable, ActionListener,MouseListener,MouseMotionListener, ...
                    ItemListener{
        /**
         *
         */
        public final static BasicStroke thickStroke = new BasicStroke(2.0f);
        private static final long serialVersionUID = -4770547484708439236L;
        Dimension area;//Groesze der Zeichenflaeche
        Gravity particle;//Integrator
        //Widgets
        PotentialFenster potentialfenster;
        //TextField mouseP;
        //JLabel timeLabel;      //record time elapse
```

```
        JComboBox InitialCombo;
        JComboBox lawCombo;
        JPanel p;
        JTextField tf;
        //Einstellregler fuer die Geschwindigkeit
        static final int SPEED_MIN = 100;
        static final int SPEED_MAX = 10000;
        static final int SPEED_INIT = 1000;     //initial frames per second
        JSlider speed;
        //Dateispeicherung
        JCheckBox cbsavefile;
        File numericdateiname;
        FileWriter numericdatei;
        //JLabel lhead;
        public final static Color BG_COLOR = new Color(0xEA,0xEA,0xEA);
//      Color BG_COLOR=Color.cyan;//lightGray;
        int yOffset=50;
        int XP[]=new int[3],YP[]=new int[3];
        String parameter[]={"Reset","C1","C2","C3","L1","L2","L3","L4","MSG1","Preset1 ...
            ","Preset2","Preset3",
                        "Preset4","Preset5","Preset6","Preset7","Preset8","Preset9"}; ...
                            //Parameter

        String defaultParameter[] = {"Reset","Fixed Initial Velocity","Fix Angular  ...
            Momentum","Arbitrary Initial Conditions",
                        "Normal Mode","Law Of Equal Areas","save to file ...","void"," ...
                            Initial Condition","1", "2", "3","4","5","6",
                        "7","8","9"};
        potential pot;
        private int version = 0;
        public String sVersion;
        /**
         * Initialisieren der TwoBodyMotion Klasse
         *      @author Fu-Kwun Hwang, Michael Wuenscher, Sebastian Eiser
         */
        public void init(){
                String tmpparam;
                try {sVersion = System.getProperty("java.runtime.version");

                } catch (SecurityException e) {
                        // handle exception
                        System.out.println("Failed to get SystemProperty. Switching to ...
                            default.");
                        sVersion = "unknown (Couldn't read system property)";
                }
                int dotPos = 0;
                dotPos = sVersion.indexOf(".");
                if(dotPos!=0) {
                        try {
                                version = Integer.parseInt(sVersion.substring(dotPos ...
                                    +1,dotPos+2));
                        }

                         catch (NumberFormatException e) {
                                //  handle exception
                                System.out.println("Warning: NumberFormatException in  ...
                                    TwoBodyMotion.java (init())");
                                version = -2;
                                // display warning
                        }

                } else if (version <= -1){
                        System.out.println("Warning: Java Runntime Version not  ...
                            detected. This might cause problems.");
                        // display warning?!
                } else {
                        System.out.println("Version: "+sVersion+" detected: "+version) ...
                            ;
                }


                for(int i=0;i<parameter.length;i++){//Einlesen der Parameter
```

```
                    if((tmpparam=getParameter(parameter[i]))!=null)
                            parameter[i]=new String(tmpparam);
                    else {
                            parameter[i]=defaultParameter[i];
                    }
            }
            //BG_COLOR=getBackground();
            setBackground(BG_COLOR);
            JPanel p=new JPanel();
            p.setBackground(getBackground());
            p.setOpaque(true);
            delay=50;//Animationsgeschwindigkeit in ms?
            speed =new JSlider(JSlider.HORIZONTAL, SPEED_MIN, SPEED_MAX,  ...
                SPEED_INIT);
            speed.setMajorTickSpacing(SPEED_MAX/5);
            speed.setPaintTicks(true);

            p.add(InitialCombo=new JComboBox());

            for(int i=1;i<4;i++)
                    InitialCombo.addItem(parameter[i]);

            countLaw=InitialCombo.getSelectedIndex();
            p.add(lawCombo=new JComboBox());
            for(int i=4;i<6;i++)lawCombo.addItem(parameter[i]);
            JButton resetButton=new JButton(parameter[0]);
            p.add(resetButton);

            //p.add(timeLabel=new JLabel(parameter[10]+"000.00 s,"+parameter[11]+" ...
                 1000 "));
            cbsavefile = new JCheckBox(parameter[6]);
            cbsavefile.addItemListener(this);
            p.add(cbsavefile);


            add("North",p);

            addMouseListener(this);
            addMouseMotionListener(this);

            resetButton.addActionListener(this);
            InitialCombo.addActionListener(this);
            lawCombo.addActionListener(this);


            particle = new Gravity(); //Integratorobjekt anlegen
            double potfakpot[] = new double[2];
            potfakpot[0] = -GM;
            potfakpot[1] = -1;

            pot = new potential(potfakpot,1);

            potentialfenster = new PotentialFenster(this);
            potentialfenster.setBackground(BG_COLOR);
//          if (version >=5)
//                  potentialfenster.setAlwaysOnTop(true); // this works only with ...
    version 5 or higher!
            reset(true);

        }
        int countLaw;



        // application code
        Image offImage,fgImage,panelImage;
        Graphics g,g0,gn,gf;
        int size=3,size2=2*size;
        int xc,yc,xs,ys;
        int nX;//Stuetzstellen in X-Richtung
        int x0,y0;//Feldmittelpunkt Offsetbefreit!!
        int chy;//Abstand zwischen den Zeilen
```

```
        double angularMomentum;
        FontMetrics fm;
        int law=1;

        /**
 * @author Fu-Kwun Hwang, Michael Wuenscher
         * @param resetButton
         */
        public void reset(boolean resetAnfangsbedingungen){
                running=false;
                area=getSize();
                area.height-=yOffset;
                countLaw=InitialCombo.getSelectedIndex();
                if (g == null) {//Nur beim ersten mal!!
                        offImage = createImage(area.width, area.height);
                        fgImage = createImage(area.width, area.height);
                        g0=getGraphics();
                        g0.setColor(Color.black);
                        gf=fgImage.getGraphics();
                        g=gn= offImage.getGraphics();
                        x0=area.width/2;
                        y0=area.height/2;
                        XP[0]=x0;
                        YP[0]=y0;
                        nX=x0-size;//Anzahl der Pixel in einer Haelfte minus dem Rand ...
                            der groesze size
                        fm=gn.getFontMetrics();
                        chy=fm.getHeight();
                }
                //double beta=String2d(tf.getText());
                //particle.init((double)(xm0-x0),0.,0.,Vcst*(y0-ym0),GM,beta*GM);
                if (resetAnfangsbedingungen){
                        xm=xm0=x0+25*size2;
                        ym=ym0=y0-yOffset;
                }else{
                        xm=xm0;
                        ym=ym0;
                }

                angularMomentum=(double)((xm0-x0)*(y0-ym0));
                clear();
                init_potential(nX);
                //TODO mehrere Koeffizienten Potenzen Paare

                potentialfenster.init(xm-x0,angularMomentum,GM,mass,pot);
                dragging=true;
                mouseReleased(new MouseEvent(this,0,0,0, xm,ym+yOffset,0,false));
                XP[2]=xm;
                YP[2]=y0;
                running=true;
        }

        int potential[];
        double potential0[],potential2[];
        //alpha = -GM (wird im Integrator neg. gemacht)
        double GM = 1.e6,
        mass= 1.e4 / GM;
        double K=GM*mass; // GM*mass
        int X1[],X2[];

        /**
         * Berechnen der Potentialarrays zum Plotten des Potentials<p>
         * @author Fu-Kwun Hwang, Michael Wuenscher
         * @param n Anzahl der Berechnungspunkte
         */
        void init_potential(int n){//Diskretisierung des Potentials
                // TODO Zeichnung des Potentials anpassen
                potential=new int[n];
                potential2=new double[n];
                potential0=new double[n];

                X1=new int[n];
```

```
                X2=new int[n];
                for(int i=0;i<n;i++){
                        potential0[i]=-K/(i+size);
                        potential[i]=y0-(int)potential0[i];
                        X1[i]=n-i;      // n=area.width/2-size
                        X2[i]=n+i+size2;
                }
        }

        /**
         * Zeichnen des Koordinatensystems
         * @author Fu-Kwun Hwang, Michael Wuenscher
         */
        void init_plot(){
                xc=x0;
                yc=y0;
                g.setColor(Color.gray);
//              x-Achse
                g.drawLine(0,yc,xc-size,yc);
                g.drawLine(xc+size,yc,area.width,yc);
//              y-Achse
                g.drawLine(xc,0,xc,yc-size);
                g.drawLine(xc,yc+size,xc,area.height);
                drawBall(xc,yc,Color.red);//Grosze Masse
                drawArrow(g,xm,y0,(double)(ym-y0),Math.PI/2,Color.red);//Startpfeil?? ...
                    sieht man nicht?
        }


// implements Runnable
        // animation code
        Thread animThread;
        long startTime=0,lastTime;
        long delay,delta;

        //  This starts the threads.

        /* (non-Javadoc)
         * Starten des Threads und des Applets
         * @see java.applet.Applet#start()
         * @author Fu-Kwun Hwang
         */
        public void start(){
                time=0.;
                dirty=false;
                //clear();
                gf.drawImage(offImage, 0, 0, this);
                g=gf;
                drawPotential(xm0,ym0,Color.green);
                g=gn;
                //Start animating!
                if (animThread == null) {
                        animThread = new Thread(this);
                        animThread.start();
                        //Remember the starting time. of thread
                        lastTime=startTime = System.currentTimeMillis();
                }
        }

        /* (non-Javadoc)
         * Stoppen des Threads und der Anwendung
         * @see java.applet.Applet#stop()
         *
         * @author Fu-Kwun Hwang
         */
        public void stop() {
                //Stop the animating thread.
                animThread = null;
        }

        /* (non-Javadoc)
```

```
         * Routine in der die Integrationsschleife am leben gehalten wird
         * @see java.lang.Runnable#run()
         * @author Fu–Kwun Hwang, Michael Wuenscher
         */
        public void run() {
                //Just to be nice, lower this thread's priority
                Thread.currentThread().setPriority(Thread.MIN_PRIORITY);
                //This is the animation loop.
                while (Thread.currentThread() == animThread) {
                        //Advance the animation frame. with delta time
                        delta=System.currentTimeMillis()–lastTime;
                        if(running)
                                advanced(delta/(double)speed.getValue()); //1000.);// ...
                                    Eigentliche Integration
                        lastTime+=delta;
                        startTime += delay;
                        try {
//                              Begrenzen der Berechnung; Maximal alle delay times ...
    einen delta Schritt
                                Thread.sleep(Math.max(0,startTime–System. ...
                                    currentTimeMillis()));
                        } catch (InterruptedException e) {
                                break;
                        }//catch (SecurityException e){
                        //      break;
                        //}
                }
        }
        boolean running=false;
        double sign=1.;
        int xxx,yyy;
        int size3=2,size4=2*size;
        boolean dirty=true;

        void drawTrace(){
                drawTrace(false);
        }

        /** Zeichnen der Bahn
         * @param status
         * @author Fu–Kwun Hwang, Michael Wuenscher, Sebastian Eiser
         */
        void drawTrace(boolean status){
                xxx=x0 + (int) particle.yout[0];
                yyy=y0 – (int) particle.yout[1];
                if(status){
                        XP[1]=XP[2]=xxx;
                        YP[1]=YP[2]=yyy;
                        return;
                }

                if(law==2){
                        if((int)(time * 3.) % 2 ==0){// 3/2 von der Zeit??
                                gf.setColor(Color.gray);
                                XP[1]=XP[2];
                                YP[1]=YP[2];
                                XP[2]=xxx;
                                YP[2]=yyy;
                                gf.fillPolygon(XP,YP,3);
                                //gf.drawLine(x0,y0,xxx,yyy);
                        }else{
                                XP[2]=xxx;
                                YP[2]=yyy;
                                if(dirty&& xxx>x0 && yyy<y0){
                                        gf.drawImage(offImage, 0, 0, this);
                                        dirty=false;
                                }else if(xxx<x0)dirty=true;
                        }
                }

                gf.setColor(Color.gray);
                ((Graphics2D)gf).setStroke(thickStroke);
```

```
                gf.drawLine(xxx,yyy,xxx,yyy);//Zeichenbefehl der Elipse Trajektorie
                ((Graphics2D)gf).setStroke(new BasicStroke());

                if(shn){
                        //writeText(xxx,yyy);
                        gf.setColor(BG_COLOR);
                        gf.fillRect(1,1,100,3*chy);
                        gf.setColor(Color.blue);
                        if(law==1) {

                                gf.drawString(parameter[8]+":",5,2*chy);
                                gf.drawString("r = <"+Integer.toString((int) xm0-x0) + ...
                                    ",0> m",5,3*chy);
                                gf.drawString("v = <0,"+Integer.toString((int) y0-ym0 ...
                                    )+"> m/s",5,4*chy);
                                gf.drawString("|L| = "+Integer.toString((int)  ...
                                    angularMomentum)+" Js",5,5*chy);
                                gf.drawLine(5,3*chy+1,10,3*chy+1);
                                gf.drawLine(5,4*chy+1,10,4*chy+1);
                                gf.drawLine(9,5*chy+1,14,5*chy+1);
                        }//else if(law==2){
                        gf.drawString("t="+d2String(time)+" s",5,chy);
                //}
                }
                shn=!shn;

                g0.drawImage(fgImage, 0, yOffset, this);//Bild wird gezeichnet
                //yOffset muszte nicht beachtet werden, da es beim zeichnen des Bildes ...
                     beachtet wurde
                yyy+=yOffset;
                y0+=yOffset;

                g0.setColor(Color.blue);
                ((Graphics2D)g0).setStroke(thickStroke);
                if (yyy<yOffset+1) {
                        g0.drawLine(x0,y0,xxx,yOffset+2);
                }else{
                        g0.drawLine(x0,y0,xxx,yyy);
                }
//              if(law==1){
//                      int xx=x0+rc;
//
//              }else if(law==2){
//                      //g0.drawString("t="+d2String(time)+"s",5,chy+yOffset);
//              }
                //Zeichnen des Zweiten Koerpers
                g0.setColor(Color.black);
                if (yyy>yOffset+3) {
                        g0.fillOval(xxx-3,yyy-3,6,6);   // trajectory
                }
                if(law==4){
                        int r=(int)Math.sqrt(particle.yout[0]*particle.yout[0]+
                                particle.yout[1]*particle.yout[1]);
                        if((r<nX)&& r>=size){//Punkte im Potentialverlauf
                                g0.setColor(Color.blue);
                                g0.fillOval(x0+r-1,potential[r-size]-1+yOffset,3,3);  ...
                                    // energy trace
                                g0.fillOval(x0-r-1,potential[r-size]-1+yOffset,3,3);  ...
                                    // energy trace
                        }
                }

                //Geschwindigkeitspfeil
                double l=Math.sqrt(particle.yout[2]*particle.yout[2]+particle.yout[3]* ...
                     particle.yout[3]);
                //Betrag der Geschwindigkeit
                double theta=Math.atan(-particle.yout[3]/particle.yout[2])+Math.PI;
                if(particle.yout[2]>0)theta=Math.PI+theta;
                drawArrow(g0,xxx,yyy,sign*l,theta,Color.red);//Geschwindigkeitspfeil
                y0-=yOffset;
                ((Graphics2D)g0).setStroke(new BasicStroke());
```

```
}

double count=200.,time;
boolean shn=false;

/**
 * Hier wird die Numerische Integration durchgefuehrt
 * @param dt Zeit die Gerechnet werden soll (Mit 200 Rechenpunkten)
 * @author Fu-Kwun Hwang, Michael Wuenscher
 */
void advanced(double dt){
        time+=dt;
        dt/=count;
        for(int i=0;i<count;i++)        particle.move(dt);
        schreibedatei();
        repaint();
}
/**
 * @author Michael Wuenscher
 *
 */
void schreibedatei(){
        if (numericdatei!=null){
                try{
                        numericdatei.write(particle.getzeile(time));
                }catch(IOException iox){
                        cbsavefile.setSelected(false);
                }
        }
}
/**
 * @author Fu-Kwun Hwang
 *
 */
void drawBall(int xi, int yi,Color ci){
        Color c=g.getColor();
        g.setColor(ci);
        g.fillOval((int)xi-size3,yi-size3,size4,size4);
        g.setColor(c);
}
/**
 * @author Fu-Kwun Hwang
 *
 */
void drawBall(int xi, int yi){
        Color c=g.getColor();
        g.setColor(BG_COLOR);
        g.fillOval((int)xs-size3,ys-size3,size4,size4);
        g.setColor(c);
        g.fillOval((int)xi-size3,yi-size3,size4,size4);
        xs=xi;
        ys=yi;
}

int xm,ym,xm0=xm+1,ym0; //Anfangsimpulse?
boolean dragging=false;

/**
 * @param x
 * @param y
 * @return
 * @author Fu-Kwun Hwang
 *
 */
int process(int x,int y){
        //int value;
        switch(countLaw){
        case 0:
                angularMomentum=(x-x0)*(y0-ym0);
                return ym0;
        case 1:
```

```
                return y0-(int)(angularMomentum/(x-x0));
        case 2:
                angularMomentum=(x-x0)*(y0-y);
                return y;
        }
        return 0;
}
// Listener Implementierung

/** (non-Javadoc)
 * @see java.awt.event.ActionListener#actionPerformed(java.awt.event. ...
     ActionEvent)
 *
 * @author Fu-Kwun Hwang, Michael Wuenscher, Sebastian Eiser
 */
public void actionPerformed(ActionEvent ev) {//Behandlung der Ereignisse des ...
    Benutzers
        //JOptionPane.showMessageDialog(this,"Action");

        if (PotentialPanel.ENTER_TEXT_FIELD.equals(ev.getActionCommand())) {
                System.out.println("enter");
        }

        if (PotentialPanel.APPLY_COMMAND.equals(ev.getActionCommand())) {
                //JOptionPane.showMessageDialog(this,"Action");
                applyingNewPot();
                return ;
        }

        if (PotentialPanel.COMBO_BOX_CHANGED.equals(ev.getActionCommand())) {
                JComboBox cb = (JComboBox)ev.getSource();
                String selectedPresent = (String)cb.getSelectedItem();
        potentialfenster.potp.setPresent(selectedPresent);

        applyingNewPot();

        }
        if (ev.getSource() == potentialfenster){
                potentialfenster.setVisible(false);
                return ;
        }

        if (ev.getSource() instanceof JButton) {
                //JOptionPane.showMessageDialog(this,"Action");
                String label = ((JButton)ev.getSource()).getText();
                if(label.equals(parameter[0]))
                        reset(true);
                return ;
        }
        if(ev.getSource()==lawCombo){
                law=lawCombo.getSelectedIndex()+1;
                if(law==2){
                        dirty=true;
                        drawTrace(dirty);
                }else gf.drawImage(offImage, 0, 0, this);
                if(law==4){
                        g=gf;
                        drawPotential(xm0,ym0,Color.green);//law3(Color.yellow ...
                            );
                        g=gn;
                }
                //else law3(Color.lightGray);
                return ;
        }
        if(ev.getSource()==InitialCombo){
                        countLaw=InitialCombo.getSelectedIndex();
                        return ;
        }
        /*if(ev.getSource() == tf){
                reset(false);//tf.setText("Value: ");
                return ;
        }*/
```

```
        //return true;
}

/**
 * @author Michael Wuenscher
 *
 */
void applyingNewPot() {

        pot = new potential(potentialfenster.potp.getCoefficients(), ...
            potentialfenster.potp.getOrder());
        pot.setscale(GM);
        reset(false);
}

boolean rightClick=false;
/**
 * @author Fu-Kwun Hwang, Michael Wuenscher, Sebastian Eiser
 *
 */
public void mousePressed(MouseEvent e){
        //JOptionPane.showMessageDialog(this,"Action");
        int x=e.getX(), y=e.getY();
        if((y-=yOffset)<0)return ;//Nur wenn es im Zeichenfeld ist

        if(e.getButton()!=MouseEvent.BUTTON1){//"Right Click, ";
                rightClick=true;
        }else rightClick=false;

        if(animThread!=null)stop();//Anhalten der Berechnung (Abfrage eher ...
            unnoetig?)

        if(Math.abs(x-xm0)<5.){
                //Loeschen alter Pfeil
                drawArrow(g,xm0,y0,(double)(ym-y0),Math.PI/2,this.BG_COLOR);
                y=process(x,y);
                dragging=true;
                g=gf;
                //drawPotential(xm0,ym0,BG_COLOR);
                //draw_elipse(xm0,ym0);
                xm0=xm=x;
                ym0=ym=y;
                drawArrow(g,xm,y0,(double)(ym-y0),Math.PI/2,Color.blue);
                //drawPotential(xm,ym,Color.green);
                //TODO Laeuft nicht im Browser
                //potentialfenster.setAlwaysOnTop(true);

                //if (version >=5)
                        //potentialfenster.setAlwaysOnTop(true); // this works ...
                            only with version 5 or higher!

                //potentialfenster.toFront();
                potentialfenster.setAnfangswerte(xm-x0,angularMomentum);
                repaint();
        }

        running=!running;//Keine Berechnungen im Thread
        //if(running)lhead.setText(parameter[8]);
                //else lhead.setText(parameter[9]);
        //if(running) writeText(x,y);

        return ;
}
/**
 * @author Fu-Kwun Hwang, Michael Wuenscher
 *
 */
public void mouseDragged(MouseEvent e){
        int x=e.getX(), y=e.getY();
        if((y-=yOffset)<0)return ;

        if (x < x0+2 || x > x0+getWidth()/2 -5) {
```

```
                //System.out.println("Mouse(X) out of range: x= "+x);
                return; // abort here
            }

            if (y > y0+2 || y < 1) {
                //System.out.println("Mouse (Y) out of range: y= "+y);
                return; // abort here
            }

            if(dragging){
                g.drawImage(offImage, 0, 0, this);
                y=process(x,y);
                xm=x;
                ym=y;
                drawArrow(g,xm,y0,(double)(ym-y0),Math.PI/2,Color.blue);
                //drawPotential(xm,ym,Color.green);
                //draw_elipse(xm,ym);
                //g.drawLine(x0+size,y0,area.width,y0);
                //g.drawLine(0,y0,x0-size,y0);
                potentialfenster.setAnfangswerte(xm-x0,angularMomentum);
                //potentialfenster.toFront();
                potentialfenster.repaint();
                repaint();
            }
            //writeText(x,y);
            return ;
        }

        boolean up;

        //int ax,ay;
        /**
         * @author Fu-Kwun Hwang, Michael Wuenscher, Sebastian Eiser
         *
         */
        public void mouseReleased(MouseEvent e){
            int x=e.getX(), y=e.getY();
            if((y-=yOffset)<0)return;

            if (x < x0+2 || x > x0+getWidth()/2 -5) {
                //System.out.println("Mouse (X) out of range: x= "+x);
                x=x0+25*size2; // ....
            }
            if (y > y0+2 || y < 1) {
                //System.out.println("Mouse (Y) out of range: y= "+y);
                y = y0-yOffset;
            }

            if(dragging){
                dragging=false;
                gf.drawImage(offImage, 0, 0, this);
                //clear();
                //drawPotential(xm0,ym0,BG_COLOR);
                y=process(x,y);
                //ym-=yOffset;
            //      init_plot();
                g=gn;
                drawArrow(g,x,y0,(double)(y-y0),Math.PI/2,Color.blue);
                //double beta=String2d(tf.getText());
                //pot.potentialfakpot[2]=beta*GM;
                particle.init((double)(x-x0),0.,0.,(y0-y),pot);//GM,beta*GM);
                //drawPotential(x,y,Color.green);
                //draw_elipse(x,y);
                potentialfenster.setAnfangswerte(xm-x0,angularMomentum);
                potentialfenster.repaint();
//                  TODO Laeuft nicht im Browser
                if (version >=5)
                        potentialfenster.setAlwaysOnTop(false); // this works ...
                            only with version 5 or higher!

                //potentialfenster.setAlwaysOnTop(false);
                //potentialfenster.toBack();
```

```
                /* if(law==3){
                        int xn=x1+(int)(R*R*R/3.e4),yn=y1-(int)(T*T/1.5);
                        g.setColor(Color.green);
                        g.fillOval(xn,yn-size3,3,3);
                        g.setColor(Color.gray);
                }*/
                if((x-x0)*(y0-y)>0)sign=1.; else sign=-1.;
                xm0=x;
                xm=-1; // keep arrow
                ym0=y;
        }
        if(!rightClick)running=!running;//Wieder Rueckgaengig machen wenn es ...
            kein Rechtsklick war
        //writeText(x,y);
        drawTrace();
        start();
        return ;
}
/**
 * @author Fu-Kwun Hwang
 * ...
 */
public boolean mouseMove(Event e, int x, int y){
        //if(!running) writeText(x,y-yOffset);
        return true;
}
public void mouseClicked(MouseEvent arg0) {
        // TODO Auto-generated method stub
}
public void mouseEntered(MouseEvent arg0) {
        // TODO Auto-generated method stub
}
public void mouseExited(MouseEvent arg0) {
        // TODO Auto-generated method stub
}
public void mouseMoved(MouseEvent arg0) {
        // TODO Auto-generated method stub
}
double T;

/**
 * Zeichnet das Potential mit der Energie
 * @param x
 * @param y
 * @param ci
 *
 * @author Fu-Kwun Hwang, Michael Wuenscher
 *
 */
void drawPotential(int x,int y,Color ci){
        //if(law==4||dragging)draw_potential();//Zeichnen der Unteren aechste

        double cst;
        Color c=g.getColor();
        g.setColor(ci);

        //Scheint nichts benutzt zu werden (zum Loeschen?)
        if(ci==BG_COLOR)drawEnergy(x-x0,ci);

        cst=(y0-y)*(x-x0);
        cst=mass*cst*cst/2.;
        for(int i=0;i<nX;i++){
                potential2[i]=potential0[i]+cst/((i+size)*(i+size));
                potential[i]=y0-(int)potential2[i];
                if(potential[i]<0)potential[i]=-1;
        }
        if(law==4||dragging){//Zeichnen der Oberen aeste
                g.drawPolyline(X1,potential,nX);
                g.drawPolyline(X2,potential,nX);
        }
        //Energie Zeichnen und Elipsenkoordinanten berechnen
```

```
        if(ci!=BG_COLOR && y!=y0)drawEnergy(x-x0,Color.red);

        int xx=(int)Math.abs(x-x0);
        double a=-K/(2.*potential2[xx-size]);

        T=2.*Math.PI*Math.sqrt(a*a*a*mass/K);

        /* if(a>0)timeLabel.setText(parameter[10]+d2String(T)+"s,"+parameter ...
             [11]+String.valueOf(R));
                else timeLabel.setText(parameter[10]+parameter[12]);//,area. ...
                     width-100,20);
        */
        int xn=x1+(int)(R*R*R/3.e4),yn=y1-(int)(T*T/1.5);

        if(dragging){
                if(ci==BG_COLOR){
                        gn.setColor(Color.gray);
                        gn.fillOval(xn,yn-size3,3,3);
                }
                g.fillOval(xn,yn-size3,3,3);

                int b=(int)(angularMomentum*T/(Math.PI*R));
                g.setColor(Color.gray);
                g.drawOval(r0,y0-b/2,x-r0,b);

                gn.drawLine(xn,yn,xn,yn);
        }else if(g==gn){
                if(ci==Color.green)
                        g.setColor(Color.black);
                g.fillOval(xn,yn-size3,3,3);
        }
        g.setColor(c);
}
/**
 * @author Fu-Kwun Hwang
 *
 */
void draw_elipse(int x,int y){
        Color c=g.getColor();
        int xi = x -x0,xx;
        if (xi<0)xi=-xi;
        xi-=size;
        double ptmp=potential2[xi];
        for(xx=0; (potential2[xx]>ptmp) &&  xx<xi ;xx++);
        if(xx==xi){
                for(xx=nX-1;potential2[xx]>ptmp && (xx>xi);xx--);
        }
        xi+=size;
        xx+=size;
        double cst=(y0-y)*(x-x0);
        cst=mass*cst*cst/2.;
        for(int i=0;i<nX;i++){
                potential2[i]=potential0[i]+cst/((i+size)*(i+size));
                potential[i]=y0-(int)potential2[i];
                if(potential[i]<0)potential[i]=-1;
        }

        //int xx=(int)Math.abs(x-x0);
        r0=x0-xx;
        R=(xx+xi)/2;
        rc=xi-xx;
        double a=-K/(2.*potential2[xi-size]);

        T=2.*Math.PI*Math.sqrt(a*a*a*mass/K);

        //int b=(int)(angularMomentum*T/(Math.PI*R));

        g.setColor(Color.gray);
        //if(dragging)g.drawOval(r0,y0-b/2,x-r0,b);
        g.setColor(c);
}
/**
```

```
 * @author Fu-Kwun Hwang
 *
 */
String d2String(double d){
        float d2=(float)((int)(10.*d)/10.);
        String str=String.valueOf(d2);
        if(str.indexOf(".")==-1)str+=".0";
        return str;
}
/**
 * @author Fu-Kwun Hwang
 *
 */
double String2d(String text){
        double tmp=Double.parseDouble(text);
        if (Double.isNaN(tmp))return 0.;
        return tmp;
}

int rc,R,r0;
/**
 * @author Fu-Kwun Hwang, Michael Wuenscher
 *
 */
void drawEnergy(int xi,Color ci){
        int xx,yy;
        double ptmp;
        if(xi<0)xi=-xi;
        xi-=size;
        ptmp=potential2[xi];
        for(xx=0; (potential2[xx]>ptmp) &&  xx<xi ;xx++);
        if(xx==xi){
                for(xx=nX-1;potential2[xx]>ptmp && (xx>xi);xx--);
        }
        xx+=size;
        xi+=size;
        Color c=g.getColor();
        g.setColor(ci);
        yy=y0-(int)ptmp;
        R=(xx+xi)/2;
        rc=xi-xx;
        if(law==4 || dragging){
                g.drawLine(x0+xx,yy,x0+xi,yy);
                g.drawLine(x0-xx,yy,x0-xi,yy);
                if(dragging){
                        g.drawLine(x0+xx,yy+5,x0+xx,yy);
                        g.drawLine(x0+xi,yy,x0+xi,y0);
                        g.drawLine(r0=x0-xx,yy,x0-xx,y0);
                        if(ci!=BG_COLOR)g.setColor(Color.gray);
                        g.drawString("r=("+String.valueOf(xx)+"+"+String. ...
                             valueOf(xi)+")/2="+String.valueOf(R),x0+5,20);
                //gf.drawString(d2String(R*R*R/(T*T)),0,20);
                }
        }
        g.setColor(c);
}
/**
 * @author Fu-Kwun Hwang, Michael Wuenscher, Sebastian Eiser
 */
void drawArrow(Graphics gs,int x, int y, double length, double theta,Color ci) ...
    {
        Color c=gs.getColor();
        gs.setColor(ci);
        length*=0.5;
        int xx=x+(int)(length*Math.cos(theta)),yy=y+(int)(length*Math.sin( ...
             theta));
        if (yy>yOffset+5 && y >yOffset+5){

        gs.drawLine(x,y,xx,yy);
        length*=0.15;
        gs.drawLine(xx,yy,xx-(int)(length*Math.cos(theta+Math.PI/6.)),
                yy-(int)(length*Math.sin(theta+Math.PI/6.)));
```

```
        gs.drawLine(xx,yy,xx-(int)(length*Math.cos(theta-Math.PI/6.)),
                yy-(int)(length*Math.sin(theta-Math.PI/6.)));
        }else{
                if (y >yOffset+5) {
                        length=(int)Math.abs(Math.round((y-yOffset)/Math.sin( ...
                            theta)));
                        xx=x+(int)(length*Math.cos(theta));
                        yy=y+(int)(length*Math.sin(theta));

                        gs.drawLine(x,y,xx,yy);
                }
        }
        gs.setColor(c);
}



int x1,y1,w1,h1;//Ursprung und Dimension fuer das Law3 Diagramm
/**
 * @author Fu-Kwun Hwang, Sebastian Eiser
 */

void clear(){//Warum clear?
        //if(g!=null){
                //Erase the previous image.
                gn.setColor(getBackground());
                gn.fillRect(0, 0, area.width, area.height);
                init_plot();
                gn.setColor(Color.gray);
                //gn.drawRect(0,0,area.width-1,area.height-1);
                x1=20;
                y1=area.height-20;
                w1=area.width/2-40;
                h1=area.height/2-40;

                law3(Color.gray);
                gn.drawString("Your Java Runtime Version is: "+sVersion,5,area ...
                    .height-gn.getFontMetrics().getHeight()*2-5);
                gn.drawString("copyright 1997, Fu-Kwun Hwang  NTNU",5,area. ...
                    height-5);
                gn.drawString("copyright 2006, Michael Wuenscher & Sebastian  ...
                    Eiser TU Berlin",5,area.height-gn.getFontMetrics(). ...
                    getHeight()-5);
}
/**
 * @author Fu-Kwun Hwang, Michael Wuenscher#
 */

void law3(Color ci){
        //gn.setColor(ci);
        //gn.drawLine(x1,y1,x1+w1,y1);
        //gn.drawLine(x1,y1,x1,y1-h1);
        //gn.drawString("T*T",0,area.height/2+chy);
        //gn.drawString("R*R*R",w1/2,y1+chy);
        //if(ci==Color.lightGray){
        //      int dx=x1+(int)(h1*(R*R*R/3.e4)/(T*T/1.5));
        //      for(int i=0;i<8;i++){
        //              gn.drawLine(x1,y1-5+i,dx+i,0);
        //      }
        //}
        gf.drawImage(offImage, 0, 0, this);
}
/**
 * @author Fu-Kwun Hwang
 */
public void paint(Graphics gs){
        super.paint(gs);
        update(gs);
}

public void update(Graphics gs){
        //super.update(gs);
```

```
                if(gs==gf){
                        gs.drawImage(fgImage, 0, yOffset, this);
                }else{
                        int r=(int)Math.sqrt(particle.yout[0]*particle.yout[0]+
                                        particle.yout[1]*particle.yout[1]);
                        potentialfenster.setTeilchen(r);
                        potentialfenster.repaint();
                        drawTrace();

                }
        }
        /**
         * @author Michael Wuenscher, Sebastian Eiser
         */
        public void stateChanged(ChangeEvent arg0) {
                // TODO Auto-generated method stub

                if (arg0.getSource()==speed){
                        delay = speed.getValue();
                        return;
                }
        }
        /**
         * @author Michael Wuenscher
         */
        public void itemStateChanged(ItemEvent arg0) {
                // TODO Auto-generated method stub
                if (arg0.getSource() == cbsavefile && arg0.getStateChange()==ItemEvent ...
                    .SELECTED){
                        JFileChooser dateidialog = new JFileChooser();

                        if (dateidialog.showSaveDialog(this) == JFileChooser. ...
                            APPROVE_OPTION){
                                numericdateiname = dateidialog.getSelectedFile();

                                        try {
                                                numericdatei = new FileWriter( ...
                                                    numericdateiname);
                                                numericdatei.write("Zeit\tx\ty\tVx\tVy ...
                                                    \n");
                                        }catch(IOException iox){
                                                cbsavefile.setSelected(false);
                                                numericdatei = null;
                                                numericdateiname = null;
                                        }
                        }else cbsavefile.setSelected(false);
                        return;
                }
                if (arg0.getSource() == cbsavefile && arg0.getStateChange()==ItemEvent ...
                    .DESELECTED){
                        try{
                                numericdatei.flush();
                        }catch(IOException iox){}
                        numericdatei = null;
                        numericdateiname =null;
                }
        }
}
```

Listing 2: Implementierung das zweiten Fensters

-5mm

```
/**
 *
 * This file (PotentialFenster.java) is part of the Two Body Motion project.
 *
 * required files: potential.java
 *
 * used by: TwoBodyMotion.java
 *
 * ----------------------------------------------------------------------
 *
```

```
 *
 *
 * @author Michael Wuenscher, Sebastian Eiser
 * modified by:
 */

import java.awt.*;
import javax.swing.*;
import java.awt.event.*;

/**
 * @author Michael Wuenscher, Sebastian Eiser
 */
class PotentialFenster extends JFrame implements ComponentListener{
        /**
         *
         */
    final static float dash1[] = {5.0f};
    final static BasicStroke dashed = new BasicStroke(1.0f,
                                                      BasicStroke.CAP_BUTT,
                                                      BasicStroke.JOIN_MITER,
                                                      10.0f, dash1, 0.0f);
    public final static BasicStroke thickStroke = new BasicStroke(2.0f);
    public final static BasicStroke  thinStroke = new BasicStroke(0.5f);

    public final static Color BG_COLOR = new Color(0xEA,0xEA,0xEA);
    private final static int LINE = 17;
        private static final long serialVersionUID = 1L;
        Dimension flaeche;
        Insets raender;
        int yOffset;
        int potential[], effpotential[],drehimpulspotential[];
        float potentialK[], effpotentialK[],drehimpulspotentialK[];
        float drehimpuls,drehimpulsq=2;
        int x0=400,y0=300;//Ursprung
        int dx,dy;//Zeichenbreite
        int size=3,size2=size*2;
        int X1[],nX=100;
        float K=1000,masse;
        int radius,radius2;
        float radiusK;
        boolean initialisiert=false;
        double potfaktoren;
        potential pot;
        PotentialPanel potp;
        int aktuellerradius;
        Image statisch;
        Graphics Gstatisch;
        ////////////////////////////////////////////////////////////////////////

        String potFensterMessage = "";
        Rectangle outOfRange;
        Rectangle drawMinimal;

//      dreieckIcon test;
        PotentialFenster(ActionListener _action){
                this.setTitle("Potential");
                this.setSize(400,600);
                this.setLocation(650,0);
                setBackground(BG_COLOR);
                //setOpaque(true);
                potp = new PotentialPanel(getSize());

                potp.setOpaque(true);
                getContentPane().setLayout(new BorderLayout());
                potp.setActionListener(_action);
                getContentPane().add(potp,BorderLayout.NORTH);
                //this.disableEvents(WindowEvent.WINDOW_STATE_CHANGED);

                addComponentListener(this);

        }
```

```
        public void init(int _radius,double _drehimpuls,double _GM,double _masse, ...
            potential _pot){
                //test = new dreieckIcon(true);
                //JPanel p=new JPanel();
                //p.add(new JLabel("Hallo"));
                //,SwingConstants.NORTH);
                K=(float)(_GM*_masse);
                masse = (float)_masse;
                //pot = new potential(potp.getCoefficients(),potp.getOrder());
                pot = new potential(_pot);
                pot.setscale(masse);
                //

                this.setVisible(true);
                setKoordinaten();
                setAnfangswerte(_radius,_drehimpuls);
//              setAlwaysOnTop(true);
                //toFront();
                //repaint();
        }
        private int EnergyToSIUnit (int wert) {

                return Math.round(wert / masse);
        }

        public void setTeilchen(int _radius){
                aktuellerradius = _radius;
        }

        void setKoordinaten(){
                flaeche = getSize();
                raender = getInsets();
                yOffset = potp.getHeight();
                y0=(flaeche.height-raender.top-raender.bottom-yOffset)/2 + raender.top ...
                     +yOffset;
                dy= flaeche.height - y0 -raender.top;//in positive y-Richtung
                x0= 2*raender.left;
                dx= flaeche.width - x0 - raender.right;

                if (statisch ==null){
                        statisch = createImage(flaeche.width,flaeche.height);
                        Gstatisch = statisch.getGraphics();
                }
                initialisiert = true;

//              unless the minimum is in that area, we have to rescale the PotFenster ...
    :
                outOfRange = new Rectangle(5, 5, (flaeche.width-10), (flaeche.height ...
                    +100));
                drawMinimal = new Rectangle(5,40, (flaeche.width-10), 100);
        }

        void setAnfangswerte(int _radius ,double _drehimpuls){
                drehimpuls = (float)(_drehimpuls * masse);
                drehimpulsq = drehimpuls * drehimpuls;
                radius = Math.abs(_radius);
                setKoordinaten();
                init_potential(dx);
        }

        public void paint(Graphics g) {// Hier kann man Zeichnen
                   super.paint(g);
                   //JOptionPane.showMessageDialog(this,"Show"); GeneralPath curveTo
                   g.setColor(Color.black);

                   draw_achsen(g);
                   draw_potential(g);
                   draw_energie(g);
                   draw_teilchen(g);
                   draw_customText(g);
        }
```

```
private void draw_customText(Graphics gPaint) {

        if(potFensterMessage.startsWith("Warning:"))
                gPaint.setColor(Color.red);
        else
                gPaint.setColor(Color.darkGray);

        gPaint.drawString(potFensterMessage,15,flaeche.height-10);


}
public void update(Graphics g){
        //Löschen
        Gstatisch.setColor(getBackground());
        Gstatisch.fillRect(0,0,flaeche.width,flaeche.height);
        //Zeichnen
        paint(Gstatisch);
        //Auf den Bildschirm
        g.drawImage(statisch,0,0,null);
}

void init_potential(int n){
        // TODO Zeichnung des Potentials anpassen

        double fa = 1; //0.000001;
        potentialK=new float[n];
        effpotentialK=new float[n];
        drehimpulspotentialK=new float[n];
        X1=new int[n];

        // fill arrays with potential data
        for(int i=0;i<n;i++) {
                potentialK[i]=-(float)pot.gety(i+1)*(float) fa;
                drehimpulspotentialK[i]=K*masse*drehimpulsq/(2*(i+1)*(i+1))*( ...
                    float) fa;
                effpotentialK[i] = potentialK[i] + drehimpulspotentialK[i];
                X1[i]=i+1 + x0;
        }

        int nMin=0;
        int potMin = 0;

        int u=1;
        double dMin = effpotentialK[0];
        for (u=1; u < n; u++) {

                if (effpotentialK[u] < dMin) {
                        dMin = effpotentialK[u];
                        nMin = u;
                }

        }
        //System.out.println("min = " + Double.toString(dMin)+" r= "+nMin);
                        //nMin = u;
                        //potMin = Math.round(effpotentialK[u]); ...
                            ////////////////////////////////////////
                        //bPotType = true;

        if (nMin != n-1) {
                potMin = Math.round(effpotentialK[nMin]);
                potFensterMessage = "Minimum at: r= "+nMin+" m, E= " + ...
                    EnergyToSIUnit(potMin)+" J";
                System.out.println(potFensterMessage);

        } else {
                potFensterMessage = "Warning: no Minimum detected.";
                System.out.println(potFensterMessage);
        }

        if (outOfRange.contains(nMin, potMin+y0)) {
                //System.out.println("well inside...");
```

```
                } else {
                        potFensterMessage = "Warning: Minimum is out of range (r="+ ...
                            nMin+", E="+EnergyToSIUnit(potMin)+").";
                        System.out.println(potFensterMessage);
                }
                if (nMin < 10) {
                        potFensterMessage = "Warning: Minimum is close to origin (r="+ ...
                            nMin+", E="+EnergyToSIUnit(potMin)+").";
                        System.out.println(potFensterMessage);
                }

                if (potMin > 0) {
                        y0 = flaeche.height-100;
                        transformCoordinates(n);
                } else {
                        y0 = (flaeche.height-raender.top-raender.bottom-yOffset)/2 + ...
                            raender.top +yOffset;
                        transformCoordinates(n);
                }
        }


        void transformCoordinates(int n) {

                potential=new int[n];
                effpotential=new int[n];
                drehimpulspotential=new int[n];

                for (int i = 0; i < n; i++) {
                        potential[i]=y0-Math.round(potentialK[i]);
                        drehimpulspotential[i]=y0-Math.round(drehimpulspotentialK[i]);
                        effpotential[i]=y0-Math.round(effpotentialK[i]);
                        if( potp.contains(X1[i],Math.round(potentialK[i])) ) {
                                //System.out.println("x="+X1[i]+ "y="+potentialK[i]);
                        }


                }
        }

        void draw_spitze(Graphics gf,int spitzex,int spitzey,boolean rotation){
                if(rotation){
                        gf.drawLine(spitzex-5,spitzey-3,spitzex,spitzey);
                        gf.drawLine(spitzex,spitzey,spitzex-5,spitzey+3);
                }else{
                        gf.drawLine(spitzex-3,spitzey+5,spitzex,spitzey);
                        gf.drawLine(spitzex,spitzey,spitzex+3,spitzey+5);
                }
        }

        void draw_achsen(Graphics gf) {
//              x-Achse

                gf.drawLine(raender.left,y0,flaeche.width-raender.right-raender.left, ...
                    y0);
                draw_spitze(gf,flaeche.width-raender.right-raender.left,y0,true);
                gf.drawString("r",flaeche.width-raender.right-raender.left-10,y0+gf. ...
                    getFontMetrics().getHeight());
                //y-Achse
                gf.drawLine(x0,yOffset+raender.top,x0,flaeche.height-raender.bottom);
                draw_spitze(gf,x0,yOffset+raender.top,false);
                gf.drawString("E",x0+5,gf.getFontMetrics().getHeight()+raender.top+ ...
                    yOffset);

                //((Graphics2D)gf).setStroke(dashed);
                gf.drawRect(flaeche.width-128,flaeche.height-4*LINE-8,120,52);
                gf.setColor(Color.red);
                gf.setFont(new Font("Helvetica", Font.BOLD, 14));
                gf.drawString("Physical pot.",flaeche.width-125,flaeche.height-3*LINE ...
                    -10);
                gf.setColor(new Color(0xED,0x89,0x32));
                gf.drawString("Centrifugal pot.",flaeche.width-125,flaeche.height-2* ...
```

```
            LINE-10);
        gf.setColor(Color.blue);
        gf.drawString("Effective pot.",flaeche.width-125,flaeche.height-LINE ...
            -10);

        //((Graphics2D)gf).setStroke(new BasicStroke());

    }

    void draw_potential(Graphics gf){

        // GC

        ((Graphics2D)gf).setStroke(thickStroke);
        gf.setColor(Color.red);
        /*for (int i = 0; i < potential.length-1; i++) {
            if ( drawMinimal.contains(X1[i+1],potential[i+1])) {
                ((Graphics2D)gf).setStroke(dashed);
                gf.drawLine(X1[i],potential[i],X1[i+1],potential[i+1]) ...
                    ;

                ((Graphics2D)gf).setStroke(thickStroke);
            }else
                gf.drawLine(X1[i],potential[i],X1[i+1],potential[i+1]) ...
                    ;

        }       */      gf.drawPolyline(X1,potential,dx);
        gf.setColor(new Color(0xED,0x89,0x32));
        /*for (int i = 0; i < drehimpulspotential.length-1; i++) {
            if (drawMinimal.contains(X1[i+1],drehimpulspotential[i+1])) {
                ((Graphics2D)gf).setStroke(dashed);
                gf.drawLine(X1[i],drehimpulspotential[i],X1[i+1], ...
                    drehimpulspotential[i+1]);

                ((Graphics2D)gf).setStroke(thickStroke);
            }else
                gf.drawLine(X1[i],drehimpulspotential[i],X1[i+1], ...
                    drehimpulspotential[i+1]);

        }
        //*/ gf.drawPolyline(X1,drehimpulspotential,dx);
        gf.setColor(Color.blue);
        /*for (int i = 0; i < effpotential.length-1; i++) {
            if (drawMinimal.contains(X1[i+1],effpotential[i+1])) {
                ((Graphics2D)gf).setStroke(dashed);
                gf.drawLine(X1[i],effpotential[i],X1[i+1],effpotential ...
                    [i+1]);

                ((Graphics2D)gf).setStroke(thickStroke);
            }else
                gf.drawLine(X1[i],effpotential[i],X1[i+1],effpotential ...
                    [i+1]);

        }
        // */ gf.drawPolyline(X1,effpotential,dx);
        ((Graphics2D)gf).setStroke(new BasicStroke());

    }

    void draw_energie(Graphics gf){

        float energieK=effpotentialK[radius-1]; // -1
        int energie =effpotential[radius-1];
        int i=radius;
        for(i=radius-2;i>-1;i--){
            if(effpotentialK[i]>=energieK)break;
        }
        if (i==radius-1){
            for(i=radius;i<dx;i++){
                if(effpotentialK[i]>=energieK)break;
            }
            i--;
```

```
                }else i++;

                radius2=i;
//              Radiusbeschriftung
                ((Graphics2D)gf).setStroke(dashed);
                gf.drawLine(x0+radius,y0,x0+radius,energie);
                String text=Integer.toString(radius);
                gf.drawString(text +"␣m",x0+radius-gf.getFontMetrics().stringWidth( ...
                    text),y0+gf.getFontMetrics().getHeight());
//              Energielinie Stroke
                gf.drawLine(x0/*+radius*/,energie,x0+radius,energie);
                text= Integer.toString(EnergyToSIUnit(Math.round(energieK)));
                gf.drawString(text+"␣J",x0,energie - 5);
                ((Graphics2D)gf).setStroke(new BasicStroke());
        }

        void draw_teilchen(Graphics gf){
                Color speicher = gf.getColor();
                gf.setColor(Color.black);
                if (aktuellerradius<dx) gf.fillOval(x0+aktuellerradius-3,effpotential[ ...
                    aktuellerradius]-3,6,6);
                gf.setColor(speicher);
        }

        public void componentResized(ComponentEvent e){
                        //JOptionPane.showMessageDialog(this,"Hallo");
                        if (e.getComponent()==this){
                                //JOptionPane.showMessageDialog(this,"Hallo");
                                statisch = null;
                                setKoordinaten();
                                init_potential(dx);
                                repaint();
                        }
                        //repaint();
        }

        public void componentMoved(ComponentEvent e){
                //JOptionPane.showMessageDialog(this,"Move");
                repaint();
        }
        public void componentHidden(ComponentEvent e){
                //JOptionPane.showMessageDialog(this,"Hidden");
                //repaint();
        }

        public void componentShown(ComponentEvent e){
                //JOptionPane.showMessageDialog(this,"Show");
                //repaint();
        }


}
```

Listing 3: Panel der Formel

-5mm

```
/**
 * This file (PotentialPanel.java) is part of the Two Body Motion project.
 *
 * required files: ArrowIconSeb.java, upDownPanel.java, FormulaPanel.java
 *
 * used by: TwoBodyMotion.java
 *
 * ----------------------------------------------------------------------
 *
 * This class (derived from JPanel) provides entering a formula (also referred as ...
     polynome of nth order)
 * out-of-the-box.
 * Warning: the construction of the object will fail, if you dont provide enough space ...
     specified by a
 * parameter during construction
```

```
 *
 *
 * The panel automatically calculates how many FormulaPanels fit into the provided ...
     space. it is possible
 * to resize the panel. It takes care of resetting and adding FormulaPanels.
 *
 * From outside, several functions are offered:
 *
 * * SetActionListener() allows the 'apply' button to trigger events outside its panel
 * * GetOrder() returns the current number of open FormulaPanels
 * * getCoefficients() returns an array of GetOrder()*2 double values representing the ...
      coefficients and
 *   power.
 *
 *
 * The main routine at the end of the file is for testing purposes only.
 *
 * @author Sebastian Eiser, August 2006
 * modified by:
*/
import java.awt.Dimension;
import java.awt.GridBagConstraints;
import java.awt.GridBagLayout;
import java.awt.GridLayout;
import java.awt.Insets;
import javax.swing.*;
import java.awt.event.*;


public class PotentialPanel extends JPanel  implements ActionListener {

        private final static int BUTTONS_WIDTH = 41;// width of the buttons, measured
        private final static int FORMULA_PANEL_WIDTH = 72;// width of the FormulaPanel
        private final static int FORMULA_PANEL_HEIGHT = 70;// hight of the ...
            FormulaPanel
        private final static int BORDER_SIZE = 20;
        private final static int FRAME_PADDING = -10;// experimental, but works
        private final static int PLUS_LABEL_WIDTH = 21; // also the '+' sign has a ...
            width (measured)
        private final static int PADDING = 0;// when using floating layout, padding is ...
             added automatically
        // between two objects; if we use GridbagLayout, no padding is added

        private final static int ABSOLUTE_MAX_ORDER = 10;// we need a maximum only for ...
             the generation of the array

        public final static String APPLY_COMMAND = "PotentialPanel_apply"; // return ...
            value
        public final static String COMBO_BOX_CHANGED = "PotentialPanel_ComboBoxChanged ...
            "; // return value
        public final static String ENTER_TEXT_FIELD = "upDownPanel_Enter_Pressed"; // ...
            return value

        private final static double[] KEPLER_PARAMETER = {1,-1,-1};
        private final static double[] PERTUBATED_KEPLER_PARAMETER = {2,-1,-1,-7,-2};
        private final static double[] HOOKE_PARAMETER = {1,0.0001,1};
        private final static double[] HARM_OSC_PARAMETER = {1,0.0000005,2};
        private final static double[] HARM_OSC_MODIFIED_PARAMETER =  ...
            {2,0.0000005,2,-1,-1};
        private final static double[] REPULSIVE_PARAMETER = {1,2,-1};
        private final static double[] QUANTENDOT_POT_PARAMETER =  ...
            {2,0.000001,2,-0.01,-1};
        private final static double[] PRESET_1_PARAMETER = {2,0.000001,2,1.1,-1};

        private final static double[][] PARAMETER = {KEPLER_PARAMETER,  ...
            PERTUBATED_KEPLER_PARAMETER, HOOKE_PARAMETER,
                HARM_OSC_PARAMETER, HARM_OSC_MODIFIED_PARAMETER, REPULSIVE_PARAMETER,  ...
                    QUANTENDOT_POT_PARAMETER,
                PRESET_1_PARAMETER};

        private final static String KEPLER_OPTION = "Kepler";
```

```
    private final static String MOD_KEPLER_OPTION = "Pertubated Kepler";
    private final static String CONSTANT_OPTION = "Constant Force Pot.";
    private final static String OSC_OPTION = "Harmonic Oscillator";
    private final static String MOD_OSC_OPTION = "Mod. Harmon. Osc.";
    private final static String REPULSIVE_OPTION = "Repulsive Potential";
    private final static String QUANT_OPTION = "\"Quantendot\"-Pot.";
    private final static String PRESET_1_OPTION = "Preset 1";

    private int maxFittingOrder = 4,// how many order fit into the frame? (default ...
        : 4)
                        nOrder = 1,// current no of order
                        savedFrameWidth=1;// holds the width of the superior ...
                            frame

    private JPanel buttonPanel;
    private JButton reset, addOrder, apply;
//      create array of ABSOLUTE_MAX_ORDER Panels, but dont create elements:
    private FormulaPanel[] formula = new FormulaPanel[ABSOLUTE_MAX_ORDER];

    private JLabel plusLabel[] = new JLabel[ABSOLUTE_MAX_ORDER -1];

    private String[] presetStrings = { KEPLER_OPTION, MOD_KEPLER_OPTION , ...
        CONSTANT_OPTION, OSC_OPTION,
                MOD_OSC_OPTION, REPULSIVE_OPTION, QUANT_OPTION, ...
                    PRESET_1_OPTION};
    // {order, coef1, pot1[, coef2, pot2 ...]}

    protected JComboBox presetList;
    protected JLabel legendLabel;

    // Constructor:
    public PotentialPanel (Dimension m) {
        if (m.width <= (BORDER_SIZE + FORMULA_PANEL_WIDTH + BUTTONS_WIDTH)
                    || m.height <= (FORMULA_PANEL_HEIGHT + BORDER_SIZE)) {
            System.out.println("Construction of PotentialPanel failed: The ...
                provided area is too small.");
            return; // this is a exception! (not yet implemented)
        }
        savedFrameWidth = m.width;// save this for comparison
        // construct widgets:
        addWidgets();
        formula[0] = new FormulaPanel(-1.000,-1);// create the first ...
            FormulaPanel

        setLayout(new GridBagLayout());// we only use GridBagLayout because it ...
             doesnt put padding between Objects by default.
        GridBagConstraints c = new GridBagConstraints();

        add(formula[0],c);
        add(buttonPanel,c);
        c.gridy=1; // second row
        c.gridwidth = 25; // this is a trick; it esures that presetList will ...
            always stay at LINE_END
        c.weightx = 1.0; // enable floating
        //c.anchor = GridBagConstraints.LINE_START;
        //add(legendLabel,c);
        c.anchor = GridBagConstraints.LINE_END; // place it to the right
        add(presetList,c);

        setBorder(BorderFactory.createCompoundBorder(
                        BorderFactory.createTitledBorder("Potential"),
                        BorderFactory.createEmptyBorder(0,0,0,0)));
        // the panel's minimum size
        this.setMinimumSize(new Dimension(1 * (BORDER_SIZE + ...
            FORMULA_PANEL_WIDTH)
                + BUTTONS_WIDTH, FORMULA_PANEL_HEIGHT + BORDER_SIZE));
        //this.setMaximumSize(new Dimension(0,FORMULA_PANEL_HEIGHT + ...
            BORDER_SIZE + 5)); // no limit: 0
    }

    private void addWidgets() {
```

```
        buttonPanel = new JPanel();
        buttonPanel.setLayout(new GridLayout(3,1));
        reset = new JButton("Reset");
        addOrder = new JButton("Add");
        apply = new JButton("Apply");
        // minimize padding of default button:
        reset.setMargin(new Insets(0,0,0,0));
        addOrder.setMargin(new Insets(0,0,0,0));
        apply.setMargin(new Insets(0,0,0,0));
//      reset.setActionCommand("reset"); // funktioniert das deaktiviert, oder …
    muss ich es beim
                // Initialisieren setten? -> michael bitte testen.
        reset.setActionCommand("reset");
        addOrder.setActionCommand("add");
        //apply.addActionListener(this);        // see public string …
            setActionListener(...);
        reset.addActionListener(this);
        addOrder.addActionListener(this);
        apply.setActionCommand(APPLY_COMMAND);

        buttonPanel.add(reset);
        buttonPanel.add(addOrder);
        buttonPanel.add(apply);

        presetList = new JComboBox(presetStrings);
        presetList.setSelectedIndex(0);
        //presetList.addActionListener(this);

}

private int calcWidth() {

        int nButtonWidth = buttonPanel.getWidth();// to make getWidth() return …
             a value different to 0,
                // the Panel has to exist already (after setVisible()).
        int temp = (this.getWidth() - FRAME_PADDING - nButtonWidth)
                / (formula[0].getWidth() + PLUS_LABEL_WIDTH + PADDING);
                        // round down..? hm, it works just fine.
        return temp;
}

public String setActionListener(ActionListener frameActionListener) {
        apply.setActionCommand(APPLY_COMMAND);
        apply.addActionListener(frameActionListener);
        presetList.setActionCommand(COMBO_BOX_CHANGED);
        presetList.addActionListener(frameActionListener);// add the …
            ComboBoxChanged ActionListener

        //formula[0].setActionToUpDownPanel(frameActionListener, …
            ENTER_TEXT_FIELD); // not implementet
        return APPLY_COMMAND;
}

public void actionPerformed(ActionEvent e) {

        if ("reset".equals(e.getActionCommand()))       {

                resetPanel();
                nOrder=1;
                this.setMinimumSize(new Dimension(1 * (BORDER_SIZE + …
                    FORMULA_PANEL_WIDTH)
                        + BUTTONS_WIDTH, FORMULA_PANEL_HEIGHT));

                formula[0] = new FormulaPanel(-1.001,-1);
                GridBagConstraints c = new GridBagConstraints();
                add(formula[0],c);
                add(buttonPanel,c);
                c.gridy=1; // second row
                c.gridwidth = 25; // this is a trick; it esures that …
                    presetList will always stay at LINE_END
                c.weightx = 1.0; // enable floating
                c.anchor = GridBagConstraints.LINE_END; // place it to the …
```

```
                    right
                add(presetList,c);
                 //formula[0].setUpDownPanels(1.0,-1.0);
                validate();
            }

            if("add".equals(e.getActionCommand())) {
                nOrder++;
                if(checkBeforeAdd(nOrder)== 0) { // maximum reached?
                    System.out.println("I cant add a thing!");
                    return;
                }

                this.setMinimumSize(new Dimension(nOrder * ( ...
                    FORMULA_PANEL_WIDTH + PLUS_LABEL_WIDTH) + BUTTONS_WIDTH,
                        FORMULA_PANEL_HEIGHT + BORDER_SIZE));    // the panel's ...
                            minimum size

                remove(buttonPanel);// first, remove the buttons
                formula[nOrder-1] = new FormulaPanel(-1.0, -nOrder);// create ...
                    and init a new FormulaPanel
                // create the + sign
                plusLabel[nOrder - 2] = new JLabel("<html><font Size = ...
                    \"6\">+</font></html>");

                // again, we need a GridBagConstraint object to place the '+' ...
                    sign correctly
                GridBagConstraints c = new GridBagConstraints();
                c.weighty=1; // enable y floating
                c.ipady = 15; // enlarge the y-padding slightly
                c.anchor = c.PAGE_END; // align on the bottom
                add(plusLabel[nOrder - 2],c); // add to the previous Polynome ...
                    (default is left-to-right) with a y-offset
                c.weighty=0; // reset
                c.ipady = 0; // reset
                c.anchor = c.CENTER; // reset
                add(formula[nOrder-1],c); // add the new FormulaPanel
                add(buttonPanel,c); // add the buttons to the right again
                validate(); // we need to refresh the view
            }
        }

        private void resetPanel() { // clear everything inside the 'Potential' border

             remove(presetList);
             remove(formula[0]);
             remove(buttonPanel);
             for(int i = 1; i < nOrder; i++) { // starting at 1
                 remove(formula[i]);
                 remove(plusLabel[i - 1]);
             }
             addOrder.setEnabled(true);

        }
        private int checkBeforeAdd(int request) {

            if((request-1) >= ABSOLUTE_MAX_ORDER) {
                addOrder.setEnabled(false); // disable the add button
                return 0;
            }
            //      if the width of the frame has changed or we enter here the ...
                first time:
            if (this.getWidth() != (savedFrameWidth) || request == 1) {
                // recalculate the no of FormulaPanels:
                maxFittingOrder = calcWidth();
                savedFrameWidth = this.getWidth();
            }
//           check if there is enough space:
            if(request >= maxFittingOrder) { //no more buttons will fit
                 if(request > maxFittingOrder) {
                     JOptionPane.showMessageDialog(this, "Nicht genug ...
                         Platz vorhanden.\n" +
```

```
                                "Bitte vergrößern Sie das Fenster.");
                        request--; // reset
                        return 0; //do nothing
                       }
                       // one still fits in; but as a precaution we turn off ...
                           the add button.
                       addOrder.setEnabled(false); // disable the add button
                }
        return 1;
}

public int getOrder() {
        return nOrder;
}

public double[] getCoefficients() {
        // before calling this function, call getOrder() to determine the
        // array size (2 * nOrder)
        double[] coef = new double[2 * nOrder];
        for(int i=0; i < nOrder; i++) {
                coef[2 * i] = formula[i].getCoeff();
                coef[(2 * i)+1] = formula[i].getPotenz();
        }
        return coef;
}

public void setCoefficients(int Order, double[] Coefficients) {
        // the number of object in the double array 'Coefficients' must be ...
            equal to 2*Order!!!

        if(checkBeforeAdd(Order)== 0) // maximum reached?
        {       System.out.println("Present loading not possible...");
                return;
                }
        resetPanel();
        nOrder = Order;

        // now it should be safe to construct the panel:

        //              again, we need a GridBagConstraint
        GridBagConstraints c = new GridBagConstraints();

        for (int u = 0; u < (nOrder - 1); u++) {
                plusLabel[u] = new JLabel("<html><font Size = \"6\">+</font></ ...
                    html>");
        }
        for (int u = 0; u < nOrder; u++) {
                formula[u] = new FormulaPanel(Coefficients[2*u], Coefficients ...
                    [2*u +1] );
                }

        add(formula[0],c);

        for (int u = 1; u< nOrder; u++) {
                c.weighty=1; // enable y floating
                c.ipady = 15; // enlarge the y-padding slightly
                c.anchor = c.PAGE_END; // align on the bottom
                add(plusLabel[u - 1],c); // add to the previous Polynome ( ...
                    default is left-to-right) with a y-offset
                c.weighty=0; // reset
                c.ipady = 0; // reset
                c.anchor = c.CENTER; // reset
                add(formula[u],c); // add new FormulaPanel
        }

        add(buttonPanel,c);
        c.gridy=1; // second row
        c.gridwidth = 25; // this is a trick; it esures that presetList will ...
            always stay at LINE_END
        c.weightx = 1.0; // enable floating
        c.anchor = GridBagConstraints.LINE_END; // place it to the right
        add(presetList,c);
```

```
        validate(); // we need to refresh the view
    }
    public void setPresent(String selection) {

        if(selection.equals(KEPLER_OPTION)) {
            double[] coeffArray = new double[(int) KEPLER_PARAMETER[0] * ...
                2];
            for (int i=0; i < (int) KEPLER_PARAMETER[0] * 2; i++) {
                coeffArray[i] = KEPLER_PARAMETER[i+1];
            }
            setCoefficients((int) KEPLER_PARAMETER[0], coeffArray );
        }
        if(selection.equals(MOD_KEPLER_OPTION)) {
            double[] coeffArray = new double[(int) ...
                PERTUBATED_KEPLER_PARAMETER[0] * 2];
            for (int i=0; i < (int) PERTUBATED_KEPLER_PARAMETER[0] * 2; i ...
                ++) {
                coeffArray[i] = PERTUBATED_KEPLER_PARAMETER[i+1];
            }
            setCoefficients((int) PERTUBATED_KEPLER_PARAMETER[0], ...
                coeffArray );
        }
        if(selection.equals(CONSTANT_OPTION)) {
            double[] coeffArray = new double[(int) HOOKE_PARAMETER[0] * ...
                2];
            for (int i=0; i < (int) HOOKE_PARAMETER[0] * 2; i++) {
                coeffArray[i] = HOOKE_PARAMETER[i+1];
            }
            setCoefficients((int) HOOKE_PARAMETER[0], coeffArray );
        }
        if(selection.equals(OSC_OPTION)) {
            double[] coeffArray = new double[(int) HARM_OSC_PARAMETER[0] * ...
                 2];
            for (int i=0; i < (int) HARM_OSC_PARAMETER[0] * 2; i++) {
                coeffArray[i] = HARM_OSC_PARAMETER[i+1];
            }
            setCoefficients((int) HARM_OSC_PARAMETER[0], coeffArray );
        }
        if(selection.equals(MOD_OSC_OPTION)) {
            double[] coeffArray = new double[(int) ...
                HARM_OSC_MODIFIED_PARAMETER[0] * 2];
            for (int i=0; i < (int) HARM_OSC_MODIFIED_PARAMETER[0] * 2; i ...
                ++) {
                coeffArray[i] = HARM_OSC_MODIFIED_PARAMETER[i+1];
            }
            setCoefficients((int) HARM_OSC_MODIFIED_PARAMETER[0], ...
                coeffArray );
        }

        if(selection.equals(REPULSIVE_OPTION)) {
            double[] coeffArray = new double[(int) REPULSIVE_PARAMETER[0] ...
                * 2];
            for (int i=0; i < (int) REPULSIVE_PARAMETER[0] * 2; i++) {
                coeffArray[i] = REPULSIVE_PARAMETER[i+1];
            }
            setCoefficients((int) REPULSIVE_PARAMETER[0], coeffArray );
        }
        if(selection.equals(QUANT_OPTION)) {
            double[] coeffArray = new double[(int) ...
                QUANTENDOT_POT_PARAMETER[0] * 2];
            for (int i=0; i < (int) QUANTENDOT_POT_PARAMETER[0] * 2; i++) ...
                {
                coeffArray[i] = QUANTENDOT_POT_PARAMETER[i+1];
            }
            setCoefficients((int) QUANTENDOT_POT_PARAMETER[0], coeffArray ...
                );
        }
        if(selection.equals(PRESET_1_OPTION)) {
            double[] coeffArray = new double[(int) PRESET_1_PARAMETER[0] * ...
                 2];
            for (int i=0; i < (int) PRESET_1_PARAMETER[0] * 2; i++) {
                coeffArray[i] = PRESET_1_PARAMETER[i+1];
```

```
                    }
                    setCoefficients((int) PRESET_1_PARAMETER[0], coeffArray );
                }


        }

        //———————————————————————— for testing only: …
            ———————————————————— //
    private static void createAndShowGUI() {
        //Make sure we have nice window decorations.
        JFrame.setDefaultLookAndFeelDecorated(true);

        //Create and set up the window.
        JFrame frame = new JFrame("Formeleingabe");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(new Dimension(400,400));
        //Create and set up the content pane.
        PotentialPanel newContentPane = new PotentialPanel(new Dimension(400,700));
        newContentPane.setOpaque(true); //content panes must be opaque
        frame.setContentPane(newContentPane);

        //Display the window.
        //frame.pack();
        frame.setVisible(true);
    }

    public static void main(String[] args) {
        //Schedule a job for the event-dispatching thread:
        //creating and showing this application's GUI.
        javax.swing.SwingUtilities.invokeLater(new Runnable() {
            public void run() {
                createAndShowGUI();
            }
        });
    }
}
```

Listing 4: File f”ur Formelobjekte

-5mm

```
/**
 * This file (FormulaPanel.java) is part of the Two Body Motion project.
 *
 * required files: ArrowIconSeb.java, upDownPanel.java
 *
 * used by: PotentialPanel.java
 *
 * ——————————————————————————————————————
 *
 * This class is JPanel, preferrably 75 pixel wide and 63 pixel high. The size is …
     determined in oder to align
 * the components (two upDownPanels and a JLabel) correctly.
 * In general the Panel packs the components and supplies a subroutine (getValues) to …
     access the textfield
 * entries.
 * We dont need any events here, since we only stack components together.
 *
 * The main routine at the end of the file is for testing purposes only.
 *
 * @author: Sebastian Eiser, August 2006
 * modified by:
*/

import javax.swing.JPanel;
import javax.swing.JFrame;
import java.awt.*;
import java.text.NumberFormat;
import java.text.DecimalFormat;
import javax.swing.*;
import java.awt.event.*;
```

```
public class FormulaPanel extends JPanel {

        private JLabel rLabel;
        private GridBagLayout gridbag;
        private GridBagConstraints layoutConstraints;
        private upDownPanel upDownKoef, upDownPotenz;
        private Dimension formulaArea = new Dimension(83, 63);
        // constructor:
        public FormulaPanel(double initK, double initP) { // pass the intial values to ...
             construct the object

                this.setPreferredSize(formulaArea); // otherwise i'll look ugly
                gridbag = new GridBagLayout();  // most flexible layout
                layoutConstraints = new GridBagConstraints(); // to specify layout ...
                     properties
                setLayout(gridbag);      // set the GridBag Layout
        rLabel = new JLabel("<html><font Size = \"6\">r</font></html>"); // r should ...
            be big enough
        NumberFormat nFormat = NumberFormat.getInstance();
        nFormat.setMinimumFractionDigits(7);

        upDownKoef = new upDownPanel( initK, 0.1, nFormat, 40); // Koeffizienten Panel
        upDownPotenz = new upDownPanel(initP, 1,new DecimalFormat("##"),10);    // und ...
            Potenz Panel

        layoutConstraints.anchor=layoutConstraints.SOUTHWEST; // allign west
        gridbag.setConstraints(upDownKoef, layoutConstraints); // set these ...
            constraints
        add(upDownKoef);

        layoutConstraints.anchor=layoutConstraints.SOUTH; // alling to the center
        layoutConstraints.weightx = 1; // random positve number, the default (0) would ...
             glue it to upDownKoef
        layoutConstraints.ipady = 17;  // lift it a little
        gridbag.setConstraints(rLabel, layoutConstraints);
        add(rLabel);

        layoutConstraints.anchor=layoutConstraints.NORTHEAST; // exponent should ...
            reside in the upper right corner
        layoutConstraints.ipady=0; // reset
        layoutConstraints.weightx = 1;
        layoutConstraints.weighty = 1; // random positve number, to enable vertical ...
            floating
        gridbag.setConstraints(upDownPotenz, layoutConstraints);
        add(upDownPotenz);

        }
        public boolean setActionToUpDownPanel(ActionListener actionListener, String ...
            cmd) {

                upDownKoef.setActionToTextField(actionListener, cmd);
                return true;
        }
        public void setUpDownPanels(double Koef, double Pot) {
                upDownKoef.setFactor((double) Koef);
                upDownPotenz.setFactor((double) Pot);
        }


    public double[] getValues() {

        double[] temp = new double[2];
        temp[0]=upDownKoef.getFactor();
        temp[1]=upDownPotenz.getFactor();

        return temp;
    }
    public double getCoeff() {
        return upDownKoef.getFactor();
    }
```

```
    public double getPotenz() {
        return upDownPotenz.getFactor();
    }
    public Dimension getArea() {
        return formulaArea;
    }

                // ———————————————————————— for testing only: …
                    ———————————————— //
    private static void createAndShowGUI() {
        //Make sure we have nice window decorations.
        JFrame.setDefaultLookAndFeelDecorated(true);

        //Create and set up the window.
        JFrame frame = new JFrame("Formeleingabe");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(new Dimension(90,90));
        //Create and set up the content pane.
        FormulaPanel newContentPane = new FormulaPanel(-2.0, -1);
        newContentPane.setOpaque(true); //content panes must be opaque
        frame.setContentPane(newContentPane);

        //Display the window.
        //frame.pack();
        frame.setVisible(true);
    }

    public static void main(String[] args) {
        //Schedule a job for the event-dispatching thread:
        //creating and showing this application's GUI.
        javax.swing.SwingUtilities.invokeLater(new Runnable() {
            public void run() {
                createAndShowGUI();
            }
        });
    }
}
```

Listing 5: Ein Textfeld mit Button darüber und darunter

-5mm

```
/**
 * This file (upDownPanel.java) is part of the Two Body Motion project.
 *
 * required files: ArrowIconSeb.java
 *
 * used by: FormulaPanel.java
 *
 * ———————————————————————————————————————————
 *
 * this class (extended from JPanel) creates basically a JFormattedTextField with …
     arrows on top and on the
 * bottom to modify its contents. It is implemented in a JPanel and handles events on …
     its own.
 * The value of the FormattedTextField is accessible by the built-in function
 * double getFactor()
 *
 * The main routine at the end of the file is for testing purposes only.
 *
 * @author: Sebastian Eiser, August 2006
 * modified by:
*/
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.*;
import java.text.DecimalFormat;
import javax.swing.*;
import java.text.NumberFormat;
import java.util.Locale;
import java.text.ParseException;
```

```
public class upDownPanel extends JPanel
                                                    implements ActionListener {

        private JButton upFactor, downFactor;
        private JFormattedTextField factor;
        private double stepsize;
        private double  dmin = -100,
                                        dmax = 100;

        public upDownPanel(double dInit, double step, NumberFormat format, int fWidth) ...
             {
                // constructor: first number sets intial value,
                // the second sets the stepsize, third sets the format of the  ...
                    textfield
                stepsize = step;
                // create up and down icons (see file: ArrowIconSeb.java)
                Icon downButtonIcon = new ArrowIconSeb(SwingConstants.SOUTH);
        Icon upButtonIcon = new ArrowIconSeb(SwingConstants.NORTH);

        downFactor = new JButton(null,downButtonIcon);
                upFactor = new JButton(null,upButtonIcon);

                // setting up events and listeners:
                downFactor.setActionCommand("down");
                upFactor.setActionCommand("up");
                downFactor.addActionListener(this);
                upFactor.addActionListener(this);

                // minimize the default button:
                downFactor.setBorderPainted(false);
                upFactor.setBorderPainted(false);
                downFactor.setMargin(new Insets(0,0,0,0));
                upFactor.setMargin(new Insets(0,0,0,0));

                // create the TextField, apply specific format rules:
                factor = new JFormattedTextField(format);
                //       factor.addPropertyChangeListener(new PropertyChangeListener i) ...
                    ; // not implemented yet
        factor.setFocusLostBehavior(JFormattedTextField.COMMIT);         // if focus is ...
             lost, stay calm.
        // intial value:
                //factor.setText(""+ dInit);

        //factor.setColumns(abc.toString().length());
                factor.setValue(new Double(dInit));

        setLayout(new BoxLayout(this,BoxLayout.Y_AXIS)); // we use a BoxLayout here
        // adding components to the layout:
        add(upFactor);

        Dimension factorSize = factor.getMinimumSize();
        factorSize.width = fWidth;
        factor.setMinimumSize(factorSize);
        add(factor);
        add(downFactor);
        // and create a decent padding: NumberFormat
        setBorder(BorderFactory.createEmptyBorder(1,1,1,1));
        }

        public boolean setActionToTextField(ActionListener actionListener, String  ...
             command) {
                factor.setActionCommand(command);
                factor.addActionListener(actionListener);
                return true;
        }
        private double parseText(String text) {
                double temp;
                 NumberFormat nf=NumberFormat.getInstance();
                 try {

                          temp = nf.parse(text).doubleValue();
                          //System.out.println("s = "+nf.parse(factor.getText()). ...
```

```
                        doubleValue());
                } catch (ParseException ex) {
                        // TODO: handle exception
                        System.out.println("Parse␣Exception␣in␣parseText␣(upDownPanel. ...
                            java)");
                        return 0;
                }
                return temp;

        }
    public void actionPerformed(ActionEvent e) {

         if ("down".equals(e.getActionCommand())) {
                 //TODO Bad workaround
                 double temp = parseText(factor.getText());

                 if (temp >= dmax || temp <= dmin)
                         return;
                 temp-=stepsize;
                 factor.setValue(new Double(temp));

         } else if ("up".equals(e.getActionCommand())) {
                 // double temp = (Double.parseDouble(factor.getText().replaceAll ...
                     (",",".")))+stepsize;

                 double temp = parseText(factor.getText());

                 if (temp >= dmax || temp <= dmin)
                         return;
                 temp+=stepsize;
                 factor.setValue(new Double(temp));
                 //
         }
    }

    public double getFactor() {

                return parseText(factor.getText());

    }

    public void setFactor(double value) {

        //factor.setFormatterFactory( )
        //factor.getFormatter();
        factor.setValue(new Double(value));
    }
    public void setMinMax(double min, double max) {
        dmin = min;
        dmax = max;
    }

    // ------------------------------------------ for testing only: ...
        ---------------------------------- //
    private static void createAndShowGUI() {
        //Make sure we have nice window decorations.
        JFrame.setDefaultLookAndFeelDecorated(true);

        //Create and set up the window.
        JFrame frame = new JFrame("ButtonDemo");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        //Create and set up the content pane.
        NumberFormat nFormat = NumberFormat.getInstance();
        nFormat.setMinimumFractionDigits(5);
        upDownPanel newContentPane = new upDownPanel(0,0,nFormat,30);
        newContentPane.setOpaque(true); //content panes must be opaque
        frame.setContentPane(newContentPane);

        //Display the window.
        frame.pack();
        frame.setVisible(true);
```

```
    }

    public static void main(String[] args) {
        //Schedule a job for the event-dispatching thread:
        //creating and showing this application's GUI.
        javax.swing.SwingUtilities.invokeLater(new Runnable() {
            public void run() {
                createAndShowGUI();
            }
        });
    }


}
```

Listing 6: Button mit Pfeil drauf

-5mm

```
/**
 * This file (ArrowIconSeb.java) is part of the Two Body Motion project.
 *
 * required files: none
 *
 * used by: upDownPanel.java
 *
 * ----------------------------------------------------------
 * here we only create icons. the icons should contain a filled triangular.
 * overriding the paintIcon sub does the work.
 *
 *
 * copyright notes: this class is almost copied from http://java.sun.com/docs (java ...
     documentation)
 * @author Sebastian Eiser, August 2006
 * modified by:
*/
import java.awt.*;
import javax.swing.*;

public class ArrowIconSeb implements Icon, SwingConstants {
    private int width = 10;
    private int height = 5;

    private int[] xPoints = new int[4];
    private int[] yPoints = new int[4];

    public ArrowIconSeb(int direction) {
        if (direction == SOUTH) { // ehemalig links
            xPoints[0] = 0;
            yPoints[0] = 0;
            xPoints[1] = width;
            yPoints[1] = 0;
            xPoints[2] = width/2;
            yPoints[2] = height;
            xPoints[3] = width/2 -1;
            yPoints[3] = height;
        } else /* direction == NORTH */ { // nach oben
            xPoints[0] = 0;
            yPoints[0] = height;
            xPoints[1] = width;
            yPoints[1] = height;
            xPoints[2] = width/2;
            yPoints[2] = 0;
            xPoints[3] = width/2-1;
            yPoints[3] = 0;
        }
    }

    public int getIconHeight() {
        return height;
    }
```

```
    public int getIconWidth() {
        return width;
    }

    public void paintIcon(Component c, Graphics g, int x, int y) {
        if (c.isEnabled()) {
            g.setColor(c.getForeground());
        } else {
            g.setColor(Color.gray);
        }

        g.translate(x, y);
        g.fillPolygon(xPoints, yPoints, xPoints.length);
        g.translate(-x, -y);    //Restore graphics object
    }
}
```

Listing 7: Hilfsklasse zum Berechnen des Gradientens und Skalieren des Potentials -5mm

```
/**
 * This file (potential.java) is part of the Two Body Motion project.
 *
 * required files: none
 *
 * used by: TwoBodyMotion.java
 *
 * ----------------------------------------------------------
 *
 *
 *
 * @author Michael Wünscher, August 2006
 * modified by:
*/
public class potential {
        double potentialfakpot[];
        int terme;
        potential(double arg[],int _terme){
                this.terme = _terme;
                potentialfakpot = new double[2*terme];
                for(int i=0;i<2*terme;i++){
                        potentialfakpot[i]=arg[i];
                }
        }

        potential(potential arg){
                terme = arg.terme;
                potentialfakpot = new double[2*terme];
                for(int i=0;i<2*terme;i++){
                        potentialfakpot[i]=arg.potentialfakpot[i];
                }
        }

        public void setscale(double faktor){
                for(int i=0;i<terme;i++){
                        potentialfakpot[2*i]*=faktor;
                }
        }
        public double gety(double x){
                double ergebnis=0.;
                for(int i=0;i<terme;i++){
                        int index = 2*i;
                        //TODO Potentialkoefficient Negativ machen !!
                        ergebnis-=potentialfakpot[index]*Math.pow(x,potentialfakpot[ ...
                            index+1]);
                }
                return ergebnis;
        }
        /*
         * getgrad ( double Ort (je x bzw. y achse), double Betrag des Ortsvektors) ??
         */
        public double getgrad(double x,double betrag){
```

```
                double ergebnis=0.;
                for(int i=0;i<terme;i++){
                        int index = 2*i;
                        ergebnis-=potentialfakpot[index]*x*potentialfakpot[index+1]* ...
                            Math.pow(betrag,potentialfakpot[index+1]-2);
                }
                return ergebnis;
        }
}
```

Listing 8: Implementierung des Integrator

-5mm

```
/**
 * file: Gravity.java
 *
 * Beinhaltet die Klassen rk4 und Gravity.
 *
 * rk4 ist die Implementierung des Runge-Kutter-Verfahrens 4. Ordnung
 *
 * @author Fu-Kwun Hwang, Michael Wuenscher
 */

        /**
         * @author Fu-Kwun Hwang, Michael Wuenscher
         */
class Gravity extends rk4{
        // calculate one dimension motion which 'a' can change over time
        //rk4:  int n;
        //rk4:  double y[],dydx[],yout[];
        //double a=0.,dt=0.;
        //private double GM,GMB;
        potential pot;
        Gravity(){
                init(4,true);
        }

        void init(double xi,double yi,double vx,double vy,potential _pot){//double gmi ...
            ,double gmbi){
                y[0]=xi;
                y[1]=yi;
                y[2]=vx;
                y[3]=vy;
                pot = _pot;
                //GM=-gmi;
                //GMB=-gmbi;
        }

        public void derivs (double t,double y[],double dydx[]){
                double r2,r2q,r;//,theta;
                r2=y[0]*y[0]+y[1]*y[1];
                //r2q=r2*r2;
                //r2=r2*Math.sqrt(r2);//eigentlich r3
                r=Math.sqrt(r2);//eigentlich r3
                //theta=Math.atan(y[1]/y[0]);
                dydx[0]=y[2]; // Vx
                dydx[1]=y[3]; // Vy
                //Orginal
                //dydx[2]=GM*y[0]/r2; // Ax
                //dydx[3]=GM*y[1]/r2;    // Ay
                //Erweitert

                //dydx[2]=GM*y[0]/r2 + GMB*y[0]/r2q; // Ax
                //dydx[3]=GM*y[1]/r2 + GMB*y[1]/r2q;     // Ay
                dydx[2]=pot.getgrad(y[0],r);///GM*y[0]/r2 + GMB*y[0]/r2q; // Ax
                dydx[3]=pot.getgrad(y[1],r);//GM*y[1]/r2 + GMB*y[1]/r2q;         // Ay
        }

        public String getzeile(double t){
                return Double.toString(t)+"\t"+Double.toString(y[0])+"\t"+Double. ...
                    toString(y[1])+
                "\t"+Double.toString(y[2])+"\t"+Double.toString(y[3])+"\n";
```

```
        }

        public void move(double dt){
                nextmove(dt);
        }
}

// RK4
/**
 * @author Fu-Kwun Hwang
 */
abstract class rk4 {
        int n;
        public double dydx[];
        public double y[],yout[];

        public void init(int dim,boolean self){
                n=dim;
                dydx=new double[n];
                yout=new double[n];
                if(self)y=yout;
                        else    y=new double[n];
        }

        public abstract void derivs(double x,double y1[],double dydx[]);

        public void nextmove(double h,double[] yin){
                y=yin;
                core(0.,h);
        }

        public void nextmove(double h){
                core(0.,h);
        }

        private void core(double x, double h){
        /* Runge-Kutta forth-order method */
                int i;
                double xh,hh,h6;
                double dym[]=new double[n];
                double dyt[]=new double[n];
                double yt[]=new double[n];

                hh=h*0.5;
                h6=h/6.0;
                xh=x+hh;

                derivs(xh,y,dydx);
                for (i=0;i<n;i++) yt[i]=y[i]+hh*dydx[i];

                derivs(xh,yt,dyt);
                for (i=0;i<n;i++) yt[i]=y[i]+hh*dyt[i];

                derivs(xh,yt,dym);
                for (i=0;i<n;i++) {
                        yt[i]=y[i]+h*dym[i];
                        dym[i] += dyt[i];
                }

                derivs(x+h,yt,dyt);
                for (i=0;i<n;i++)
                        yout[i]=y[i]+h6*(dydx[i]+dyt[i]+2.0*dym[i]);
        }
}
```

B Original Quelltext von Fu-Kwun Hwang

Listing 9: Original Quelltext des Kepler Apletts

-5mm

```
import java.awt.*;

public class Kepler extends java.applet.Applet implements Runnable{
        Dimension area;
        Gravity particle;
        TextField mouseP;
        Label timeLabel;        //record time elapse
        Choice c;
        Color bgColor=new Color(0xC8,0xDF,0xD0);
//      Color bgColor=Color.cyan;//lightGray;
        int yOffset=50;
        int XP[]=new int[3],YP[]=new int[3];
        Choice cb;
//      String cs1="EncodingProblem",cs2="EncodingProblem",es1="planet X,Y",es2="mouse ...
    X,Y";
        Label lhead;
        boolean lang=false;
        String rts,STR[]={"Reset","C1","C2","C3","L1","L2","L3","L4","MSG1","MSG2","T" ...
            ,"R","invalid"};
        public void init(){
                for(int i=0;i<STR.length;i++){
                        if((rts=getParameter(STR[i]))!=null)
                                STR[i]=new String(rts);
                }
//              if(getParameter("lang")!=null)lang=true;
                setBackground(bgColor);
                delay=50;
                Panel p=new Panel();
//              if(lang)lhead=new Label(cs1);
//              else lhead=new Label(es1);
                p.add(lhead=new Label(STR[8]));
                p.add(mouseP=new TextField("0 , 0",10));
                p.add(c=new Choice());
                for(int i=1;i<4;i++)c.addItem(STR[i]);
/*              if(lang){
                        c.addItem("EncodingProblem");
                        c.addItem("EncodingProblem");
                        c.addItem("EncodingProblem");
                }else{
                        c.addItem("Fix Kinetic Energy");
                        c.addItem("Fix angular Momentum");
                        c.addItem("arbitrary");
                }*/
                cn=c.getSelectedIndex();
                p.add(cb=new Choice());
                for(int i=4;i<8;i++)cb.addItem(STR[i]);
/*              if(lang){
                        cb.addItem("EncodingProblem");
                        cb.addItem("EncodingProblem");
                        cb.addItem("EncodingProblem");
                        cb.addItem("EncodingProblem");
                }else{
                        cb.addItem("1st law");
                        cb.addItem("2nd law");
                        cb.addItem("3rd law");
                        cb.addItem("energy");
                }*/
                p.add(new Button(STR[0]));
                p.add(timeLabel=new Label(STR[10]+"0.00 s,"+STR[11]+" 0.0 "));
                add("North",p);
                particle=new Gravity();
                reset(false);
        }
        int cn;
        public boolean action(Event ev, Object arg) {
                if (ev.target instanceof Button) {
                String label = (String)arg;
                        if(label.equals(STR[0]))
                                reset(true);
```

```
        }else if(ev.target==cb){
                law=cb.getSelectedIndex()+1;
                if(law==2){
                        dirty=true;
                        drawTrace(dirty);
                }else gf.drawImage(offImage, 0, 0, this);
                if(law==4){
                        g=gf;
                        drawPotential(xm0,ym0,Color.green);//law3(Color.yellow ...
                            );
                        g=gn;
                }
                //else law3(Color.lightGray);
        }else if(ev.target==c)
                        cn=c.getSelectedIndex();
        return true;
}
// application code
Dimension offDimension;
Image offImage,fgImage;
Graphics g,g0,gn,gf;
int size=3,size2=2*size;
int xc,yc,xs,ys;
int nX,x0,y0,chy;
double angularMomentum;
FontMetrics fm;
int law=1;
public void reset(boolean resetButton){
        running=false;
        area=size();
        area.height-=yOffset;
        cn=c.getSelectedIndex();
        if (g == null) {
                offDimension = area;
                offImage = createImage(area.width, area.height);
                fgImage = createImage(area.width, area.height);
                g0=getGraphics();
                g0.setColor(Color.black);
                gf=fgImage.getGraphics();
                g=gn= offImage.getGraphics();
                x0=area.width/2;
                y0=area.height/2;
                XP[0]=x0;
                YP[0]=y0;
                nX=x0-size;
                fm=gn.getFontMetrics();
                chy=fm.getHeight();
        }
        particle.init((double)(xm0-x0),0.,0.,Vcst*(y0-ym0),GM);
        xm=xm0=x0+25*size2;
        ym0=ym=y0-yOffset;
        angularMomentum=(double)((xm0-x0)*(y0-ym0));
        clear();
        init_potential(nX);
        dragging=true;
        mouseUp(null,xm,ym+yOffset);
        XP[2]=xm;
        YP[2]=y0;
        running=true;
}
int potential[];
double potential0[],potential2[];
double Vcst=1.,GM=1.e6*Vcst*Vcst,mass=1.e4/GM,R3T2=GM/(4.*Math.PI*Math.PI);
// GM=r*V^2 = (100*100*100)
double K=GM*mass; // GM*mass
int X1[],X2[];
void init_potential(int n){
        potential=new int[n];
        potential0=new double[n];
        potential2=new double[n];
        X1=new int[n];
        X2=new int[n];
```

```
            for(int i=0;i<n;i++){
                potential0[i]=-K/(i+size);
                potential[i]=y0-(int)potential0[i];
                X1[i]=n-i;      // n=area.width/2-size
                X2[i]=n+i+size2;
            }
        }
        void init_plot(){
            xc=x0;
            yc=y0;
            g.setColor(Color.gray);//lack);
            g.drawLine(0,yc,xc-size,yc);
            g.drawLine(xc+size,yc,area.width,yc);
            g.drawLine(xc,0,xc,yc-size);
            g.drawLine(xc,yc+size,xc,area.height);
            drawBall(xc,yc,Color.red);
            drawArrow(g,xm,y0,(double)(ym-y0),Math.PI/2,Color.blue);
        }
        void draw_potential(){
            //gf.drawImage(offImage, 0, 0, this);
            gf.setColor(Color.green);
            for(int i=0;i<nX;i++){
                potential[i]=y0-(int)potential0[i];
                if(potential[i]<0)potential[i]=-1;
            }
            gf.drawPolygon(X1,potential,nX);
            gf.drawPolygon(X2,potential,nX);
        }
// implements Runnable
        // animation code
        Thread animThread;
        long startTime=0,lastTime;
        long delay,delta;
        //  This starts the threads.
        public void start(){
            time=0.;
            dirty=false;
            //clear();
            gf.drawImage(offImage, 0, 0, this);
            g=gf;
            drawPotential(xm0,ym0,Color.green);
            g=gn;
            //Start animating!
            if (animThread == null) {
                animThread = new Thread(this);
                animThread.start();
                //Remember the starting time. of thread
                lastTime=startTime = System.currentTimeMillis();
            }
        }
        public void stop() {
            //Stop the animating thread.
            animThread = null;
        }

        public void run() {
            //Just to be nice, lower this thread's priority
            Thread.currentThread().setPriority(Thread.MIN_PRIORITY);
            //This is the animation loop.
            while (Thread.currentThread() == animThread) {
                //Advance the animation frame. with delta time
                delta=System.currentTimeMillis()-lastTime;
                if(running)advanced(delta/1000.);
                lastTime+=delta;
                startTime += delay;
                try {
                    animThread.sleep(Math.max(0,startTime-System. ...
                        currentTimeMillis()));
                } catch (InterruptedException e) {
                    break;
                }//catch (SecurityException e){
                //      break;
```

```
                //}
        }
}
boolean running=false;
double sign=1.;
int xxx,yyy,size3=2,size4=2*size;
boolean dirty=true;
void drawTrace(){
        drawTrace(false);
}
void drawTrace(boolean status){
        xxx=x0+(int)particle.yout[0];
        yyy=y0-(int)particle.yout[1];
        if(status){
                XP[1]=XP[2]=xxx;
                YP[1]=YP[2]=yyy;
                return;
        }
        g.drawLine(xxx,yyy,xxx,yyy);
        if(law==2){
                if((int)(time*3.) %2 ==0){
                        gf.setColor(Color.gray);
                        XP[1]=XP[2];
                        YP[1]=YP[2];
                        XP[2]=xxx;
                        YP[2]=yyy;
                        gf.fillPolygon(XP,YP,3);
                        //gf.drawLine(x0,y0,xxx,yyy);
                }else{
                        XP[2]=xxx;
                        YP[2]=yyy;
                        if(dirty&& xxx>x0 && yyy<y0){
                                gf.drawImage(offImage, 0, 0, this);
                                dirty=false;
                        }else if(xxx<x0)dirty=true;
                }
        }
        gf.setColor(Color.gray);
        gf.drawLine(xxx,yyy,xxx,yyy);
        if(shn){
                writeText(xxx,yyy);
                gf.setColor(bgColor);
                gf.fillRect(1,1,100,3*chy);
                gf.setColor(Color.blue);
        if(law==1){
                int xx=x0+rc,yy;
                double l1=Math.sqrt((x0-xxx)*(x0-xxx)+(y0-yyy)*(y0-yyy)),
                        l2=2.*R-l1;
                        //l2=Math.sqrt((xx-xxx)*(xx-xxx)+(y0-yyy)*(y0-yyy));
                //gf.drawLine(xxx,yyy,xx,y0);
                gf.drawString("  l1   ="+d2String(l1),5,yy=2*chy);
                gf.drawString("  l2   ="+d2String(l2),5,yy+=chy);
                gf.drawString("l1+l2="+d2String(l1+l2),5,yy+=chy);
                //g0.setColor(Color.yellow);
                //g0.drawOval(xx-1,y0-1,2,2);
                //g0.drawString(d2String(l1)+"+"+d2String(l2)+"="+d2String(l1+ ...
                    l2),area.width/2,yOffset+chy);
        }//else if(law==2){
                gf.drawString("t="+d2String(time)+"s",5,chy);
        //}
        }
        shn=!shn;
        g0.drawImage(fgImage, 0, yOffset, this);
        yyy+=yOffset;
        y0+=yOffset;
        g0.setColor(Color.blue);
        g0.drawLine(x0,y0,xxx,yyy);
        if(law==1){
                int xx=x0+rc,yy;
                double l1=Math.sqrt((x0-xxx)*(x0-xxx)+(y0-yyy)*(y0-yyy)),
                        l2=2.*R-l1;
                        //l2=Math.sqrt((xx-xxx)*(xx-xxx)+(y0-yyy)*(y0-yyy));
```

```
                g0.drawLine(xxx,yyy,xx,y0);
                //g0.drawString("  l1   ="+d2String(l1),5,yy=chy+yOffset);
                //g0.drawString("  l2   ="+d2String(l2),5,yy+=chy);
                //g0.drawString("l1+l2="+d2String(l1+l2),5,yy+=chy);
                g0.setColor(Color.yellow);
                g0.drawOval(xx-1,y0-1,2,2);
                //g0.drawString(d2String(l1)+"+"+d2String(l2)+"="+d2String(l1+ ...
                    l2),area.width/2,yOffset+chy);
        }else if(law==2){
                //g0.drawString("t="+d2String(time)+"s",5,chy+yOffset);
        }
        g0.setColor(Color.yellow);
        g0.fillOval(xxx-2,yyy-2,4,4);  // trajectory
        if(law==4){
        int r=(int)Math.sqrt(particle.yout[0]*particle.yout[0]+
                        particle.yout[1]*particle.yout[1]);
        if((r<nX)&& r>=size){
                g0.setColor(Color.blue);
                g0.fillOval(x0+r-1,potential[r-size]-1+yOffset,3,3); // energy ...
                    trace
                g0.fillOval(x0-r-1,potential[r-size]-1+yOffset,3,3); // energy ...
                    trace
        }
        }
        double l=Math.sqrt(particle.yout[2]*particle.yout[2]+particle.yout[3]* ...
            particle.yout[3]);
        double theta=Math.atan(-particle.yout[3]/particle.yout[2]);//Math.asin ...
            (-particle.yout[3]/l);
        if(particle.yout[1]>0)theta=Math.PI+theta;
        drawArrow(g0,xxx,yyy,sign*l,theta,Color.red);
        y0-=yOffset;
}
double count=200.,time;
boolean shn=false;
void advanced(double dt){
        time+=dt;
        dt/=count;
        for(int i=0;i<count;i++)          particle.move(dt);
        repaint();
}
void drawBall(int xi, int yi,Color ci){
        Color c=g.getColor();
        g.setColor(ci);
        g.fillOval((int)xi-size3,yi-size3,size4,size4);
        g.setColor(c);
}
void drawBall(int xi, int yi){
        Color c=g.getColor();
        g.setColor(bgColor);
        g.fillOval((int)xs-size3,ys-size3,size4,size4);
        g.setColor(c);
        g.fillOval((int)xi-size3,yi-size3,size4,size4);
        xs=xi;
        ys=yi;
}
int xm,ym,xm0=xm+1,ym0;
boolean dragging=false;
int process(int x,int y){
        //int value;
        switch(cn){
        case 0:
                angularMomentum=(x-x0)*(y0-ym0);
                return ym0;
        case 1:
                return y0-(int)(angularMomentum/(x-x0));
        case 2:
                angularMomentum=(x-x0)*(y0-y);
                return y;
        }
        return 0;
}
boolean rightClick=false;
```

```
        public boolean mouseDown(Event e, int x, int y){
                if((y-=yOffset)<0)return true;
                if(e.modifiers==Event.META_MASK){//"Right Click, ";
                        rightClick=true;
                }else rightClick=false;
                if(animThread!=null )stop();
                if(Math.abs(x-xm0)<5.){
                        drawArrow(g,xm0,y0,(double)(ym-y0),Math.PI/2,Color.gray);
                        y=process(x,y);
                        dragging=true;
                        g=gf;
                        //drawPotential(xm0,ym0,bgColor);
                        xm0=xm=x;
                        ym0=ym=y;
                        drawArrow(g,xm,y0,(double)(ym-y0),Math.PI/2,Color.blue);
                        drawPotential(xm,ym,Color.green);
                        repaint();
                }
                running=!running;
                if(running)lhead.setText(STR[8]);
                        else lhead.setText(STR[9]);
                if(running)
/*              if(lang){
                        if(running)lhead.setText(cs1);else lhead.setText(cs2);
                }else{
                        if(running)lhead.setText(es1);else lhead.setText(es2);
                }*/
                writeText(x,y);
                return true;
        }
        public boolean mouseDrag(Event e, int x, int y){
                if((y-=yOffset)<0)return true;
                if(dragging){
                        g.drawImage(offImage, 0, 0, this);
                        y=process(x,y);
                        xm=x;
                        ym=y;
                        drawArrow(g,xm,y0,(double)(ym-y0),Math.PI/2,Color.blue);
                        drawPotential(xm,ym,Color.green);
                        //g.drawLine(x0+size,y0,area.width,y0);
                        //g.drawLine(0,y0,x0-size,y0);
                        repaint();
                }
                writeText(x,y);
                return true;
        }
        boolean up;
        //int ax,ay;
        public boolean mouseUp(Event e, int x, int y){
                if((y-=yOffset)<0)return true;
                if(dragging){
                        dragging=false;
                        gf.drawImage(offImage, 0, 0, this);
                        //clear();
                        //drawPotential(xm0,ym0,bgColor);
                        y=process(x,y);
                        //ym-=yOffset;
                //      init_plot();
                        g=gn;
                        drawArrow(g,x,y0,(double)(y-y0),Math.PI/2,Color.blue);
                        particle.init((double)(x-x0),0.,0.,Vcst*(y0-y),GM);
                        drawPotential(x,y,Color.green);
                        /*if(law==3){
                                int xn=x1+(int)(R*R*R/3.e4),yn=y1-(int)(T*T/1.5);
                                g.setColor(Color.green);
                                g.fillOval(xn,yn-size3,3,3);
                                g.setColor(Color.gray);
                        }*/
                        if((x-x0)*(y0-y)>0)sign=1.; else sign=-1.;
                        xm0=x;
                        xm=-1; // keep arrow
                        ym0=y;
```

```
        }
        if(!rightClick)running=!running;
        writeText(x,y);
        drawTrace();
        start();
        return true;
    }
    double T;
    void drawPotential(int x,int y,Color ci){
        if(law==4||dragging)draw_potential();
        double cst;
        Color c=g.getColor();
        g.setColor(ci);
        if(ci==bgColor)drawEnergy(x-x0,ci);
        cst=Vcst*(y0-y)*(x-x0);
        cst=mass*cst*cst/2.;
        for(int i=0;i<nX;i++){
            potential2[i]=potential0[i]+cst/((i+size)*(i+size));
            potential[i]=y0-(int)potential2[i];
            if(potential[i]<0)potential[i]=-1;
        }
        if(law==4||dragging){
            g.drawPolygon(X1,potential,nX);
            g.drawPolygon(X2,potential,nX);
        }
        if(ci!=bgColor && y!=y0)drawEnergy(x-x0,Color.red);
        int xx=(int)Math.abs(x-x0);
        double a=-K/(2.*potential2[xx-size]);
        T=2.*Math.PI*Math.sqrt(a*a*a*mass/K);
        if(a>0)timeLabel.setText(STR[10]+d2String(T)+"s,"+STR[11]+String. ...
            valueOf(R));
            else timeLabel.setText(STR[10]+STR[12]);//,area.width-100,20);
            int xn=x1+(int)(R*R*R/3.e4),yn=y1-(int)(T*T/1.5);
        if(dragging){
            if(ci==bgColor){
                gn.setColor(Color.gray);
                gn.fillOval(xn,yn-size3,3,3);
            }
            g.fillOval(xn,yn-size3,3,3);
            //if(law==3){
                int b=(int)(angularMomentum*T/(Math.PI*R));
                g.setColor(Color.gray);
                g.drawOval(r0,y0-b/2,x-r0,b);
            //}
            gn.drawLine(xn,yn,xn,yn);
        }else if(g==gn){
            if(ci==Color.green)
            g.setColor(Color.yellow);
            g.fillOval(xn,yn-size3,3,3);
        }
        g.setColor(c);
    }
    String d2String(double d){
        float d2=(float)((int)(10.*d)/10.);
        String str=String.valueOf(d2);
        if(str.indexOf(".")==-1)str+=".0";
        return str;
    }
    int rc,R,r0;
    void drawEnergy(int xi,Color ci){
        int xx,yy;
        double ptmp;
        if(xi<0)xi=-xi;
        xi-=size;
        ptmp=potential2[xi];
        for(xx=0; (potential2[xx]>ptmp) &&  xx<xi ;xx++);
        if(xx==xi){
            for(xx=nX-1;potential2[xx]>ptmp && (xx>xi);xx--);
        }
        xx+=size;
        xi+=size;
        Color c=g.getColor();
```

```
                g.setColor(ci);
                yy=y0-(int)ptmp;
                R=(xx+xi)/2;
                rc=xi-xx;
                if(law==4 || dragging){
                g.drawLine(x0+xx,yy,x0+xi,yy);
                g.drawLine(x0-xx,yy,x0-xi,yy);
                        if(dragging){
                        g.drawLine(x0+xx,yy+5,x0+xx,yy);
                        g.drawLine(x0+xi,yy,x0+xi,y0);
                        g.drawLine(r0=x0-xx,yy,x0-xx,y0);
                        if(ci!=bgColor)g.setColor(Color.gray);
                        g.drawString("r=("+String.valueOf(xx)+"+"+String.valueOf(xi)+" ...
                            )/2="+String.valueOf(R),x0+5,20);
                        //gf.drawString(d2String(R*R*R/(T*T)),0,20);
                        }
                }
                g.setColor(c);
        }
        void drawArrow(Graphics gs,int x, int y, double length, double theta,Color ci) ...
            {
                Color c=gs.getColor();
                gs.setColor(ci);
                length*=0.5;
                int xx=x+(int)(length*Math.cos(theta)),yy=y+(int)(length*Math.sin( ...
                    theta));
                gs.drawLine(x,y,xx,yy);
                length*=0.15;
                gs.drawLine(xx,yy,xx-(int)(length*Math.cos(theta+Math.PI/6.)),
                                                        yy-(int)(length*Math.sin(theta ...
                                                            +Math.PI/6.)));
                gs.drawLine(xx,yy,xx-(int)(length*Math.cos(theta-Math.PI/6.)),
                                                        yy-(int)(length*Math.sin(theta ...
                                                            -Math.PI/6.)));
                gs.setColor(c);
        }

        void writeText(int x,int y){
                mouseP.setText(String.valueOf(x-x0)+" , "+
                                String.valueOf(y0-y));
//              g.drawString(
        }

        public boolean mouseMove(Event e, int x, int y){
                if(!running)
                writeText(x,y-yOffset);
                return true;
        }
        int x1,y1,w1,h1;
        void clear(){
                //if(g!=null){
                        //Erase the previous image.
                        gn.setColor(getBackground());
                        gn.fillRect(0, 0, area.width, area.height);
                        init_plot();
                        gn.setColor(Color.black);
                        //gn.drawRect(0,0,area.width-1,area.height-1);
                        x1=20;
                        y1=area.height-20;
                        w1=area.width/2-40;
                        h1=area.height/2-40;
                        //if(law==3)law3(Color.gray);
                        //else
                        law3(Color.gray);
                        gn.drawString("copywrite 1997, Fu-Kwun Hwang  NTNU",area.width ...
                            /2+5,area.height-5);
                        //init_plot();
                        //g0.drawImage(offImage, 0, 0, this);
                //}
        }
        void law3(Color ci){
                gn.setColor(ci);
```

```
                gn.drawLine(x1,y1,x1+w1,y1);
                gn.drawLine(x1,y1,x1,y1-h1);
                gn.drawString("T*T",0,area.height/2+chy);
                gn.drawString("R*R*R",w1/2,y1+chy);
                //if(ci==Color.lightGray){
                //      int dx=x1+(int)(h1*(R*R*R/3.e4)/(T*T/1.5));
                //      for(int i=0;i<8;i++){
                //              gn.drawLine(x1,y1-5+i,dx+i,0);
                //      }
                //}
                gf.drawImage(offImage, 0, 0, this);
        }
        public void paint(Graphics gs){
                update(gs);
        }
        public void update(Graphics gs){
                if(g==gf){
                        gs.drawImage(fgImage, 0, yOffset, this);
                }else{
                        drawTrace();
                }
        }
}

//*********************************************************/
class Gravity extends rk4{
        // calculate one dimension motion which 'a' can change over time
        //rk4:  int n;
        //rk4:  double y[],dydx[],yout[];
        //double a=0.,dt=0.;
        private double GM;
        Gravity(){
                init(4,true);
        }
        void init(double xi,double yi,double vx,double vy,double gmi){
                y[0]=xi;
                y[1]=yi;
                y[2]=vx;
                y[3]=vy;
                GM=-gmi;
        }
        public void derivs (double t,double y[],double dydx[]){
                double r2,theta;
                r2=y[0]*y[0]+y[1]*y[1];
                r2=r2*Math.sqrt(r2);
                theta=Math.atan(y[1]/y[0]);
                dydx[0]=y[2]; // Vx
                dydx[1]=y[3]; // Vy
                dydx[2]=GM*y[0]/r2; // Ax
                dydx[3]=GM*y[1]/r2;     // Ay
        }
        public void move(double dt){
                nextmove(dt);
        }
}

// RK4
abstract class rk4 {
        int n;
        public double dydx[];
        public double y[],yout[];
        public void init(int dim,boolean self){
                n=dim;
                dydx=new double[n];
                yout=new double[n];
                if(self)y=yout;
                        else    y=new double[n];
        }

         public abstract void derivs(double x,double y1[],double dydx[]);
        /*
        public void next(double x,double h){ // t & dt
```

```
                core(x,h);
        }
        public double[]  next(double h){ // dt
                core(0.,h);
                return yout;
        }*/
        public void nextmove(double h,double[] yin){
                y=yin;
                core(0.,h);
        }
        public void nextmove(double h){
                core(0.,h);
        }
        private void core(double x, double h){
        /* Runge-Kutta forth-order method */
                int i;
                double xh,hh,h6;
                double dym[]=new double[n];
                double dyt[]=new double[n];
                double yt[]=new double[n];
                hh=h*0.5;
                h6=h/6.0;
                xh=x+hh;
                derivs(xh,y,dydx);
                for (i=0;i<n;i++) yt[i]=y[i]+hh*dydx[i];
                derivs(xh,yt,dyt);
                for (i=0;i<n;i++) yt[i]=y[i]+hh*dyt[i];
                derivs(xh,yt,dym);
                for (i=0;i<n;i++) {
                        yt[i]=y[i]+h*dym[i];
                        dym[i] += dyt[i];
                }
                derivs(x+h,yt,dyt);
                for (i=0;i<n;i++)
                        yout[i]=y[i]+h6*(dydx[i]+dyt[i]+2.0*dym[i]);
        }
}
```

Literatur

[1] Giesela Engeln-Müllges and Frank Uhlig. *Numerical Algorithms with C*. Springer, Berlin, 1996.

[2] Herbert Goldstein. *Klassische Mechanik*. Akad. Verlagsgesesllschaft, Wiesbaden, (aus d. engl. übers. von günter gliemann) 7. edition, 1983.

[3] Wolfgang Nolting. *Grundkurs Theoretische Physik 1. Klassische Mechanik*. Springer, Berlin, 2004.